T - 3

PENYESUAIAN BAGAN PADA FLOWCHART SEBAGAI UPAYA MENJAGA KONSISTENSI DAN KEJELASAN ALGORITMA PEMROGRAMAN KOMPUTER

Bambang Sumarno HM
Jurusan Pendidikan Matematika FMIPA UNY
bambang@uny.ac.id

Abstrak

Algoritma di dalam pemrograman komputer, merupakan rujukan langkah-langkah penyelesaian masalah, yang penyajiannya sering diabaikan. Permasalahan ini disebabkan keengganan pemrogram menuliskannya, sehingga dirasa cukup ada di ingatannya. Kondisi ini diperburuk dengan salah satu cara penyajian algoritma, yaitu flowchart yang masih terdapat beberapa permasalahan yang menjadikannya kurang komprehensif seiring perkembangan bentuk pemrograman komputer.

Terdapat tiga cara utama penyajian algoritma, yaitu: (1) verbal/kalimat pernyataan, (2) flowchart, dan (3) pseudocode. Flowchart atau Bagan Alir merupakan cara penyajian Algoritma dengan menggunakan bentuk-bentuk bangun datar tertentu yang saling terhubung membentuk sebuah aliran penyelesaian masalah.

Beberapa perbaikan komponen flowchart yang dihasilkan dari kajian ini, diantaranya: (1) penambahan bagan untuk deklarasi dan deskripsi pengenal yang diperlukan, (2) penyesuaian bagan untuk perulangan tertentu (*for - to - do*), dan (3) perubahan pemanfaatan bagan lingkaran untuk pemilihan beracuan nilai (*case - of*).

Kata Kunci: Bagan, Flowchart, Algoritma, dan Pemrograman Komputer

A. PENDAHULUAN

Ilmu Komputer sebagai sebuah ilmu telah berkembang pesat dengan tumbuhnya bidang kajian yang ada di dalamnya, seperti: Arsitektur dan Organisasi Komputer (*Computer Organisation and Architecture*), Jaringan Komputer (*Computer Network*), Pemrograman Komputer (*Computer Programming*), Kecerdasan Semu/Buatan (*Artificial Intelligence*), Penglihatan Komputer (*Computer Vision*), Robotika (Robotic), dan seterusnya. Bahkan beberapa diantaranya telah tumbuh kembang menjadi disiplin ilmu yang berdiri sendiri. Dari sekian banyak bidang kajian tersebut, Pemrograman Komputer (selanjutnya dituliskan Pemrograman) merupakan bidang kajian yang hampir tidak dapat dilepaskan dari kajian yang lainnya. Hal ini disebabkan pada setiap bagian kajian ilmu komputer selalu terdapat unsur pemrograman di dalamnya, seperti ditunjukkan pada sistem komputasi.

Lapisan Sistem Komputasi

Komputer adalah sebuah perangkat, sedangkan sistem komputasi adalah sebuah entitas dinamis yang digunakan untuk menyelesaikan permasalahan dan berinteraksi dengan lingkungannya. Sistem komputasi tersusun dari perangkat keras dan perangkat lunak, serta data yang mereka kelola. Sistem komputasi dapat diibaratkan seperti bawang, terdiri dari beberapa lapisan (layer). Setiap lapisan memainkan peran spesifik di dalam desain keseluruhan dari sistem. Terdapat 6 lapisan di dalam sistem komputasi seperti pada Gambar 1.

Dale and Lewis (2011:4-5) menerangkan; Lapisan paling dalam, Informasi (*Information*), mencerminkan cara menyajikan/mewakili informasi di dalam komputer. Penyajian ini dikelola menggunakan dijit biner, 1 dan 0. Lapisan berikutnya, Perangkat Keras (*Hardware*), terdiri dari perangkat keras fisik dari sistem komputer. Perangkat keras komputer yang dimaksud, termasuk perangkat seperti gerbang dan sirkuit yang mengontrol aliran listrik dengan cara-cara yang mendasar.

Gambar 1. Lapisan Sistem Komputasi

Sumber: Dale and Lewis

Lapisan Pemrograman (*Programming*) berhubungan dengan perangkat lunak, instruksi-instruksi yang digunakan untuk menyelesaikan komputasi dan mengelola data. Program (yang terdiri dari insruksi-instruksi tersebut) dapat diambil dalam beberapa bentuk, dapat dibentuk pada beberapa tingkat, dan dapat diimplementasikan dalam beberapa bahasa (pemrograman).

Setiap komputer memiliki sebuah Sistem Operasi (*Operating System*) untuk membantu mengelola sumber daya yang ada. Lapisan ini membantu pengguna berinteraksi dengan sistem komputer dan mengelola cara berinteraksi perangkat keras, program, dan data. Jika lapisan sistem operasi fokus pada bagaimana sistem komputer bekerja, lapisan Aplikasi (*Application*) fokus pada penggunaan komputer untuk menyelesaikan permasalahan dunia-nyata tertentu.

Komputer tidak lagi berada terisolasi di atas meja (kerja) seseorang. Teknologi komputer telah dapat digunakan untuk berkomunikasi, dan Komunikasi (*Communication*) ini merupakan lapisan mendasar dimana sistem komputer beroperasi. Hal ini menjadikan komputer dapat dihubungkan ke dalam jaringan (komputer) sehingga mereka dapat berbagi informasi dan sumber daya.

Dari lapisan yang menyusun Sistem Komputasi terlihat keberadaan Pemrograman merupakan lapisan yang memegang peranan penting sebagai penghubung lapisan perangkat keras dengan perangkat lunak yang di dalamnya terdiri sistem operasi, aplikasi, dan komunikasi. Keberadaannya yang memegang peranan penting menjadikannya sebagai kajian yang penting dan berkembang pesat di dalam Ilmu Komputer.

Algoritma Pemrograman

Pada lapisan pemrograman, tujuan (goal) dari kajiannya adalah terbentuknya program-program komputer yang bertujuan memerintah/mengakses perangkat (keras) komputer sesuai kebutuhan pengguna/manusia dengan menggunakan instruksi- instruksi yang dipahami komputer. Kumpulan instruksi ini biasanya disebut kode sumber (*source code*). Instruksi yang ada dibangkitkan menggunakan bahasa pemrograman tertentu oleh pemrogram (orang yang membuat program).

Berawal dari bahasa pemrograman yang lebih berorientasi pada mesin/komputer (yang dikenal sebagai bahasa pemrograman tingkat rendah), yaitu: bahasa mesin yang menggunakan kode biner (1 dan 0) atau bahasa assembler yang menggunakan kode semu; Berkembang menjadi bahasa pemrograman yang lebih berorientasi pada manusia (yang dikenal sebagai bahasa pemrograman tingkat tinggi), diantaranya: BASIC, Fortran, dan Pascal. Selain itu, dimunculkan juga bahasa pemrograman tingkat menengah, contohnya: bahasa C, yang masih memfasilitasi

orentasi pada mesin/komputer sehingga memungkinkan pengaksesan perangkat keras komputer secara langsung, dan fokus menyediakan antarmuka yang lebih berorientasi pada manusia untuk membungkus perintah/instruksi yang akan diberikan kepada mesin/komputer.

Seiring dengan meningkatnya kemampuan perangkat komputer dengan fitur yang lebih menjanjikan, menjadikan kebutuhan terhadap pemrograman yang dapat memberdayakan perangkat tersebut juga bermunculan. Hal ini dapat dilihat cukup banyaknya bermunculan bahasa pemrograman dengan paradigmanya yang lebih berdayaguna, diantaranya: Visual Basic, Delphi, PHP, dan Java. Masing-masing memberikan kemudahan di dalam penggunaannya ketika membuat suatu program dengan tersedianya editor yang terintegral dikenal sebagai IDE, yaitu: *Integrated Development Environment*. Adanya IDE ini, pembuatan program lebih teratur dan terbimbing, serta cukup mudah menemukan kesalahan tata tulis yang mungkin terjadi.

Keberadaan bahasa pemrograman yang beragam, di satu sisi memberikan pilihan/alternatif bagi pemrogram, tetapi di satu sisi menyebabkan terjadinya tumpang-tindih kode/sintaksis satu bahasa dengan lainnya. Meskipun tata tulis/sintaksisnya berbeda untuk masing-masing bahasa pemrograman, pembuatan rencana penyelesaian masalah yang akan dibuat programnya tetap sama. Di dalam komputasi, rencana ini disebut ***algoritma***. Asal kata Algoritma diambil dari nama Matematikawan Persia yang menulis kitab *Al Jabr wa'al-muqabala* (*rules of restoration and reduction*) di abad ke-9 (sekitar tahun 825), al Khowarizmi, adalah seperangkat aturan untuk melaksanakan beberapa perhitungan, baik secara manual (dengan tangan), atau lebih sering pada mesin/komputer (Brassard and Bratley, 1996:1).

Definisinya di dalam terminologi komputasi, algoritma adalah himpunan instruksi untuk menyelesaikan masalah atau sub-masalah pada sejumlah waktu yang terbatas dengan menggunakan sejumlah data yang terbatas (Dale and Lewis, 2011:198). Secara implisit, di dalam definisi ini adalah pemahaman bahwa instruksi yang ada tidak ambigu, yaitu setiap instruksi memiliki makna/pengertian tunggal. Pemahaman ini sering juga dikenal dengan istilah 'logis'. Sebuah algoritma adalah sebuah rencana yang tepat untuk disajikan dalam sederetan tindakan/aksi untuk sebuah pencapaian tujuan yang diharapkan (CodeWorrior,1993:74). Setiap tindakan diambil dari daftar tindakan pada data yang dipahami secara jelas/baik.

Berikut beberapa contoh algoritma: (a) mempersiapkan makan pagi, (b) memutuskan besaran biaya masuk ke bioskop, (c) menghitung luas sebuah segitiga, (d) menganti ban mobil, atau (e) bermain permainan karambol. Dapat diperhatikan bahwa setiap algoritma tersebut menentukan dua hal, yaitu:

- Setiap tindakan ditentukan dengan kata kerja, seperti mempersiapkan, menganti, dan bermain, dan
- Data yang dikenai tindakan, ditentukan dengan kata benda atau frase benda, seperti: makan pagi, bam mobil, dan permainan karambol.

Lebih jauh, CodeWorrior (1993:75) menjelaskan bahwa sebuah algoritma harus memenuhi 4 sifat sebagai berikut.

- Lengkap (*complete*); Semua tindakan harus dituliskan/didefinisikan secara tepat
- Tidak ambigu (*unambiguous*); Sekumpulan instruksi akan tidak ambigu jika hanya ada satu kemungkinan cara menginterpretasikannya
- Deterministik (*deterministic*); Jika instruksi diikuti/dikerjakan dapat dipastikan bahwa hasil yang diharapkan akan selalu dicapai.
- Terbatas (*finite*); Instruksi-instruksi harus berakhir setelah sejumlah cacah langkah/step. Kebutuhan akan pembatasan ini tidak hanya berarti berhenti setelah sejumlah cacah langkah/step, juga berarti bahwa instruksi dapat menggunakan sebuah angka terbatas dari variabel untuk mencapai hasil yang diharapkan.

Sebagai contoh, algoritma menghitung luas sebuah segitiga dengan seperangkat instruksinya seperti pada Gambar 2.

{Algoritma Menghitung Luas Segitiga}

No urut	Instruksi
1.	Tentukan nilai alas dari segitiga
2.	Tentukan nilai tinggi dari segitiga
3.	Hitung luasnya dengan perhitungan hasil kali alas dengan tinggi dibagi dua

Gambar 2. Algoritma Menghitung Luas Segitiga

Dari algoritma ini (Gambar 2) terlihat instruksi yang ada lengkap dan didefinisikan secara tepat. Mulai dari memasukkan nilai yang diperlukan (alas dan tinggi), menghitung luasnya, serta menampilkan hasilnya. Setiap instruksi dapat dimengerti dengan jelas dan tidak bermakna ganda (ambigu). Masing-masing instruksi yang diikuti/dikerjakan memberikan hasil sesuai yang diharapkan. Seperangkat instruksi yang dikerjakan dapat dipastikan berakhir setelah instruksi terakihir, yaitu: menampilkan hasil perhitungan luas segitiga.

Penggunaan/penerapanan penomoran, boleh menggunakan angka atau abjad, dari setiap intruksi untuk menunjukkan urutan pelaksanaannya. Walaupun secara umum dapat dimengerti bahwa instruksi yang berada di atas dikerjakan terlebi dahulu, penomoran ini dapat memberikan alur perhitungan nilai dari luas segitiga secara lebih teratur. Andaikan algoritma ini dikerjakan, akan diawali mulai instruksi dengan penomoran paling kecil; Jika alas diberi nilai 9 dan tinggi dengan nilai 5, maka luas akan memperoleh nilai 22,5 yaitu hasil dari (9x5)/2 yang selanjutkan akan ditampilkan nilai luas tersebut.

Walaupun algoritma pada prinsipnya dapat diterapkan untuk semua bentuk permasalah, baik dari permasalahan kehidupan sehari-hari yang sederhana, seperti mempersiapkan makan pagi, mengganti ban mobil, dan sejenisnya; Sampai permasalahan teknis yang kompleks, seperti: mekanika pada bangunan, aerodinamika pada mobil mapun pesawat, dan sejenisnya; Pada perkembangannya penggunaan algoritma diorientasikan untuk pembuatan program komputer. Perangkat instruksi yang ada di dalam algoritma akan dilakukan pengodean (*coding*) menggunakan bahasa pemrograman tertentu. Hal ini menjadikan penggunaan istilah algoritma identik dengan algoritma pemrograman. Sebelum algoritma ditranslasi ke dalam kode-kode intruksi bahasa pemrograman, cara menyajikannya menjadi salah satu bagian yang tidak dapat diabaikan.

Flowchart: Cara Penyajian Algoritma secara Grafis

Kejelasan sebuah algoritma tidak dapat dilepaskan dari cara menyajikannya. Algoritma yang di dalamnya terdapat sekumpulan langkah penyelesaian suatu permasalahan perlu disajikan secara benar agar memenuhi sifat-sifat algoritma yang baik. Algoritma mungkin muncul dalam beberapa bentuk secara spesifik yang disebut cara penyajian (*representations*) atau notasi (notations) Cara menyajikan sebuah algoritma sangat penting karena ia harus membawa secara cepat arti yang ada di dalam algoritma dengan paling sedikit usaha oleh pembacanya.Tidak satupun penyajian pantas untuk semua algoritma. Cara penyajian yang terbaik bagi salah satu algoritma, kemungkinan menjadi cara yang terburuk bagi lainnya.

Secara umum, penyajian algoritma dapat di kelompokkan menjadi 3 cara, yaitu: (a) verbal/kalimat pernyataan, (b) flowchart, dan (c) pseudocode. Cara verbal dan pseudocode penyajiannya lebih pada pada bentuk-bentuk tulisan/kode, sedangkan flowchart penyajiannya lebih pada bentuk-bentuk grafis. Gambar 2 merupakan contoh penyajian algoritma dengan cara verbal/kalimat pernyataan. Jika algoritma tersebut disajikan dengan cara pseudocode seperti pada Gambar 3.

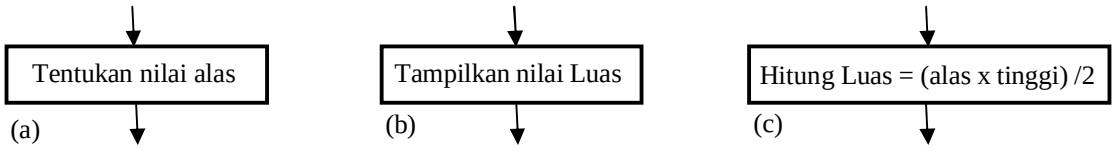
```
Program: Menghitung Luas Segitiga

  {Kode semu untuk menghitung luas segitiga dengan memberikan
  masukan nilai alas dan tingginya}

Deklarasi
    Integer  alas, tinggi;
    Real  Luas;
```

Gambar 3. Pseudocode dari Algoritma Menghitung Luas Segitiga

Pseudocode adalah kode atau tanda yang menyerupai (pseudo) untuk menjelaskan cara menyelesaikan suatu masalah. Pada pseudocode, instruksi tidak terlalu berbentuk verbal (kalimat lengkap/baku) tetapi lebih mendekati bentuk bahasa pemrograman. Kedekatan kode semu dengan salah satu bahasa pemrograman ini menjadikan keunggulan pseudocode, karena tidak perlu lagi memakan waktu ketika akan dibuat kode prorgamnya.

Flowchart adalah satu cara yang sering digunakan untuk memperlihatkan sederetan tindakan di dalam sebuah algoritma Mereka terdiri dari "kotak-kotak" terhubung oleh garis dengan tanda panah menunjukkan aliran dari sederetan tindakan. Kotak-kotak tersebut berbentuk berbeda tergantung pada apa yang mereka sajikan tindakan, keputusan, atau penegasan (CodeWorrior, 1993:107-109):

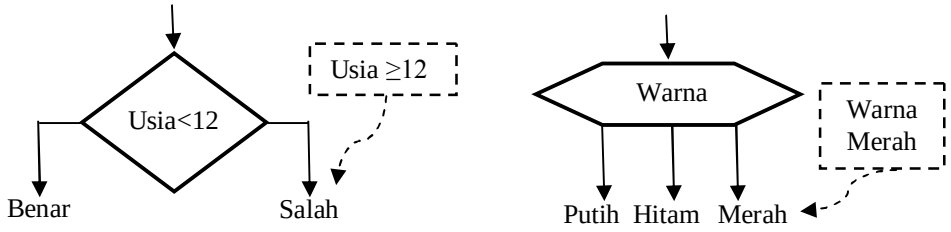
- Tindakan (*actions*) menunjukkan proses (seperti masukan, keluaran, perhitungan) dan dipresentasikan sebagai kotak persegi panjang (Gambar 4).



Gambar 4. Contoh tindakan masukan (a), keluaran (b), dan perhitungan (c)

Tindakan ada kemungkinan mengubah nilai data dimana mereka dioperasikan. Sebagai contoh, tindakan menaikkan (*incrementing*) pencacah akan merubah nilainya dengan menambah 1 pada nilai sebelumnya.

- Keputusan (*decisions*) sering menggunakan bentuk tes/uji atau pertanyaan seperti Umur<12 atau Warna? yang memiliki dua atau lebih hasil (kebanyakan tetapi tidak selalu Benar atau Salah), tergantung pada nilai yang akan diuji. Keputusan disajikan dengan belah ketupat (*diamond*) atau kotak dengan titik-titik akhir. Setiap kotak keputusan harus memiliki label tanda panah yang keluar untuk setiap hasil yang mungkin (Gambar 5). Tanda panah memandu ke tindakan berikutnya atau keputusan akan dilakukan.
- Penegasan (*assertions*) adalah pernyataan atau fakta (seperti usia ≥12 atau "Merah") terkait nilai dari variabel pada beberapa posisi di dalam flowchart. Penegasan ini ditunjukkan dengan kotak putus-putus menunjuk pada tempat dimana penegasan menyimpan nilai benar (Gambar 5).



Gambar 5. Contoh keputusan dan penegasan

Kotak yang digunakan untuk menyajikan tindakan, keputusan, dan penegasan dapat dikombinasikan untuk menyajikan algoritma yang lengkap. Pada kenyataannya, mereka dapat

dikelompokkan ke dalam bentuk tertentu yang mengindikasikan cara tindakan dan obyek munkin terhubung. Terdapat empat bentuk dasar, yaitu:

- Rangkaian/urutan (*sequence*); Pada algoritma menghitung luas segitiga menunjukan dua tindakan dalam rangkaian, satu setelah lainnya, misalnya: menentukan nilai tinggi setelah sebelumya menentukan nilai alas.
- Pemilihan (*selection*); Pada algoritma harga melibatkan pilihan tindakan pada kondisi "usia<12" adalah Benar atau Salah. Jika usia kurang dari 12, maka garis berlabel Benar akan dipilih/diikuti dan harganya Rp. 10.000, sebaliknya dengan harga Rp. 25.000.
- Pengulangan (*repetition*); Pada algoritma menjumlah sepuluh buah data melibatkan pengulangan beberapa tindakan yang tergantung pada kondisi cacah data belum melebihi sepuluh buah. Selama kondisi bernilai benar, pemasukan data dan penjumlahannya dilakukan. Kondisi cacah data>10 adalah sebagai kondisi penghentian (*termination condotion*). Tindakan memasukkan data dan menjumlahnya diketahui sebagai badan dari perulangan (*body of the loop*).
- Invokasi (*invocation*); Setiap dari tiga contoh di atas dapat diletakkan dalam sebuah kotak dan diberi label untuk referensi berikutnya. Kotak-kotak ini seperti sub-algoritma, dan mereka dapat diminta/dipakai ketika diperlukan. Sub-algoritma ditandai sebagai kotak dengan garis vertikal ganda. Invokasi dari sub-algoritma diperlakukan sebagai tindakan tunggal di dalam flowchart.

Flowchart sebagai penggambaran secara grafis dari langkah-langkah dan urut-urutan prosedur dari suatu program mengalami perkembangan. Flowchart diterapkan juga untuk menggambarkan jenis tindakan dan urutannya untuk keperluan lainnya. Sampai saat ini dikenal lima jenis flowchart, yaitu:

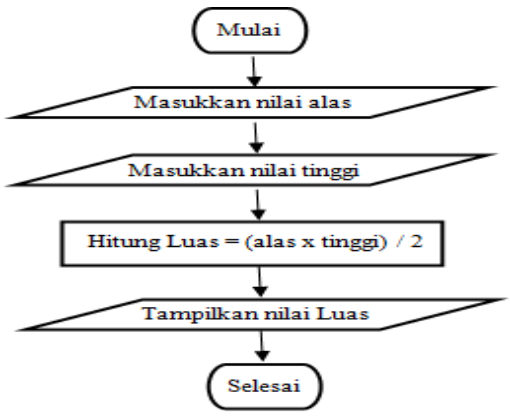
- Flowchart Sistem (*System Flowchart*); Flowchart ini menunjukkan alur kerja atau apa yang sedang dikerjakan di dalam sistem secara keseluruhan dan menjelaskan urutan dari prosedur-prosedur yang ada di dalam sistem. Dengan kata lain, flowchart ini merupakan deskripsi secara grafik dari urutan prosedur-prosedur yang terkombinasi yang membentuk suatu sistem.
- Flowchart Kertas Kerja/Dokumen (*Paperwork/Document Flowchart*); Flowchart kertas kerja/dokumen menelusuri alur dari data yang ditulis melalui sistem. Kegunaan utamanya adalah untuk menelusuri pergerakan form dan laporan sistem dari satu bagian ke bagian lain termasuk bagaimana alur form dan laporan diproses, dicatat serta disimpan.
- Flowchart Skematik (*Schematic Flowchart*); Flowchart ini mirip dengan Flowchart Sistem, tetapi tidak hanya menggunakan simbol-simbol flowchart standar, juga menggunakan gambar-gambar komputer, peripheral, form-form atau peralatan lain yang digunakan dalam sistem. Flowchart Skematik digunakan sebagai alat komunikasi antara analis sistem dengan seseorang yang tidak familiar dengan simbol-simbol flowchart yang konvensional.
- Flowchart Program (*Program Flowchart*); Flowchart Program dihasilkan dari Flowchart Sistem, merupakan keterangan yang lebih rinci tentang bagaimana setiap langkah program atau prosedur sesungguhnya dilaksanakan. Flowchart ini menunjukkan setiap langkah program atau prosedur dalam urutan yang tepat saat terjadi. Pemrogram menggunakan flowchart program untuk menggambarkan urutan instruksi dari program komputer.
- Flowchart Proses (*Process Flowchart*); Flowchart Proses merupakan teknik penggambaran rekayasa industrial yang memecah dan menganalisis langkah-langkah selanjutnya dalam suatu prosedur atau sistem.

Pada perkembangannya, simbol-simbol flowchart yang biasa dipakai adalah simbol-simbol flowchart standar yang dikeluarkan oleh ANSI dan ISO. Tabel 1 berisikan simbol-simbol yang biasa digunakan untuk Flowchart (Program), selain garis dengan tanda panah untuk menunjukkan aliran instruksi.

Tabel 1. Simbol Standar Flowchart Program

Simbol	Arti	Contoh
Titik Terminal	Awal/akhir flowchart	Mulai
Input/Output	Input data yang akan diproses atau Output data atau Informasi yang dihasilkan	Tentukan nilai alas
Proses	Proses yang dilakukan dengan melibatkan data tertentu	Hitung Luas = (alas x tinggi)/2
Keputusan	Keputusan berdasarkan hasil pengujian kondisi/syarat	Alas > 0 *salah* *benar*
Preparasi	Pemberian nilai awal (untuk perulangan dengan cacah tertentu)	Ulangi untuk i=1 s/d 10
Proses Terdefinisi	Pendefinisian sebuah proses dimana rincian tindakan berada di bagan lainnya (sub-algoritma)	Luas segitiga
Konektor	Keluar ke atau masuk dari bagan lain pada sebuah flowchart pada halaman yang sama	5 5
Konektor	Keluar ke atau masuk dari bagan lain pada sebuah flowchart pada halaman yang berbeda/berikutnya	7 7

Dengan menggunakan simbol-simbol pada Tabel 1, maka algoritma menghitung luas segitiga dapat disajikan dalam bentuk flowchart seperti pada Gambar 6.



Gambar 6. Flowchart Menghitung Luas Segitiga

Dari Tabel 1, terdaftar simbol-simbol standar flowchart (program) yang semakin presentatif. Telah ada bagan (jajar genjang) yang diperuntukkan input atau output agar berbeda dengan bagan proses/tindakan. Demikian juga untuk mengelola flowchart yang cukup kompleks, tersedia bagan proses terdefinisi untuk memecahnya menjadi beberapa flowchart yang yang lebih kecil dan khusus, yang dapat dipandang seperti sub-algoritma. Tetapi tersedia dua bagan konektor yang terkesan berlebihan.

B. PEMBAHASAN

Dari paparan di atas menunjukkan pentingnya penyusunan algoritma sebagai langkah-langkah penyelesaian suatu masalah. Langkah penyelesaian tersebut akan berupa seperangkat instruksi yang akan diberikan ke komputer untuk dikerjakan sebagaimanamestinya. Selain fokus pada instruksi-instruksi yang hendak dikerjakan, di dalam algoritma juga perlu menyertakan data yang terlibat/dikenakan instruksi tersebut. Keberadaan data yang tersurat secara jelas di dalam algoritma, akan lebih diperlukan ketika akan digunakan sebagai pedoman pembuat kode sumber dari sebuah program.

Adanya pengertian program = algoritma + struktur data, mempertegas bahwa pada setiap pemrograman tidak dapat dilepaskan dari keberadaan algoritma (yang fokus pada instruksi/proses) dan struktur dari data yang terlibat/diolah di dalamnya. Hal ini menunjukkan keberadaan keduanya, algoritma dan struktur data penting dan tidak dapat dilepaskan satu dengan lainnya di dalam pemrograman. Oleh karena itu, cara penyajian algoritma (yang di dalamnya terdapat struktur data) perlu mendapat perhatian lebih. Selain untuk kepentingan dokumentasi, cara penyajian algortima lebih utama untuk memberikan kejelasan dan arahan tahapan penyelesaian masalah yang akan dibuat koding programnya.

Penyajian algoritma yang baik, yang memenuhi kaidah-kaidah penyusunannya, merupakan bagian penting dan tidak terpisahkan dalam kesatuan proses pemrograman. Bahkan dapat diduga banyaknya kesalahan (*bug*) saat pengodean program disebabkan kurang baiknya penyajian algoritma yang digunakan rujukan. Seperti dipaparkan sebelumnya, terdapat 3 cara utama penyajian algoritma, yaitu: verbal/kalimat pernyataan, flowchart, dan pseudocode. Cara verbal/ kalimat pernyataan lebih berorientasi pada manusia, yang mana di dalamnya jarang ditemukan kode yang menjurus pada bentuk instruksi yang menyerupai bahasa pemrograman. Sebaliknya, pseudocode lebih berorientasi pada mesin dikarenakan penyajian langkah-langkah penyelesaian mendekati bentuk-bentuk instruksi dari sebuah bahasa pemrograman. Oleh karena itu, penyajian verbal banyak dilakukan oleh orang pada umumnya. Sebaliknya, pseudocode lebih diharapkan oleh pemrogram karena penerjemahan/translasi langkah penyelesaiaan ke kode sumber (program) dapat dikerjakan relatif lebih mudah dibandingkan secara verbal yang memungkin memunculkan beragam interpretasi kata/kalimat yang ada.

Adapun flowchart yang cara penyajian algoritma secara grafis dengan menggunakan bentuk bagan-bagan, menawarkan kemudahan di dua sisi. Baik bagi orang umum (bukan pemrogram) maupun pemrogram. Flowchart menolong analis dan pemrogram untuk memecahkan masalah ke dalam segmen/bagian yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian. Dibandingkan dengan cara verbal maupun pseudosode, flowchart memberikan visualisasi pembacaan instruksi secara lebih komprehensif. Hal ini dapat diberikan dengan cara penyajian langkah-langkah penyelesaian masalah diidentikkan dengan bentuk bagan tertentu. Adanya kesinergian bagan dengan instruksi yang ada dapat mempertegas apa yang harus dilakukan dan apa yang dihasilkannya.

Keunggulan ini menjadikan flowchart pilihan bagi sebagian besar orang yang sedang belajar algoritma dan pemrograman. Baik bagi pengajar untuk membelajarkan algoritma pemrograman, maupun peserta didik yang memperlajarinya. Keunikan bagan-bagan yang ada memudahkan mereka yang mengingat dan memahami instruksi yang ada, sedangkan hubungan satu bagan dengan lainnya memberikan visualisasi aliran proses yang ada. Meskipun flowchart telah menjadi cara penyajiaan yang dikenal secara luas, masih terdapat beberapa hal yang dipandang perlu dilakukan sebagai bentuk peningkatan konsistensi dan kejelasan

langkah-langkah penyelesaian masalah. Berikut tiga hal yang dipandang perlu untuk diterapkan pada flowchart.

1. Bagan untuk pendeklarasian data dan struktur/tipe datanya.

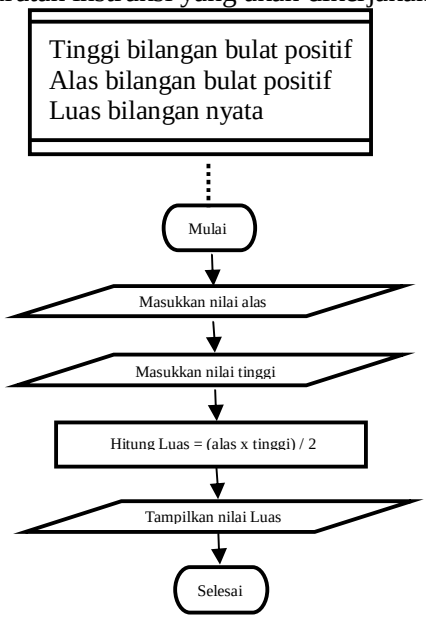
   Bagan yang ada di flowchart lebih ditujukan untuk merepresentasikan instruksi yang akan dikerjakan (oleh komputer). Keberadaan data yang dikenai instruksi diasumsikan sudah melekat/termasuk di dalamnya. Cara ini di satu sisi menawarkan fleksibilitas penentuan data, khususnya tipe/strukturnya; Tetapi kondisi ini membuka kemungkinan kesalahan pemilihan tipe/struktur yang sebaiknya.

   Bagi pemrograman (sebut saja profesional) yang sudah terbiasa membuat program dengan bahasa pemrograman tertentu, keperluan data dengan jenisnya tidak terlalu menjadi permasalahan. Lain halnya bagi mereka yang sedanga belajar (sebut saja peserta didik), kejelasan data dan tipe/strukturnya dalam bentuk deklarasi terpisah akan banyak membantu pemahaman hasil dari instruksi-instruksi yang ada. Selain itu, untuk membangun konsintensi antar cara penyajian algoritma, khususnya flowchart dengan pseudocode, terutama saat flowchart digunakan sebagai rujukan penulisan kode program. Kebutuhan terhadap deklarasi data ini diperkuat dengan kenyataan sebagian besar bahasa pemrograman memerlukan pendeklarasian data beserta tipe/strukturnya secara eksplisit/terpisah.

   Untuk keperluan ini maka ditambahkan bagan baru untuk pendeklarasian data dan tipe/strukturnya, yang berupa bagan persegi panjang dengan sisi horisontalnya berupa garis ganda ( ). Sebagai contoh flowchart untuk menyajikan algoritma untuk menghitung luas segitiga seperti pada Gambar 6, akan diperbarui seperti pada gambar 7.

   Penempatan bagan deklarasi di atas bagan mulai, dengan pertimbangan setiap data yang dikenai instruksi ada/disediakan sebelumnya. Hal ini juga untuk mempertegas konsistensi struktur dari bahasa pemrograman yang dirujuk, yaitu bahasa Pascal. Pada bahasa Pascal, semua data dideklarasikan pengenalnya di bagian deklarasi yang berada sebelum atau di atas program utama. Dengan cara seperti ini, maka blok instruksi yang berada di antara bagan mulai dan selesai dapat dipahami sebagain bagian dari program utama.

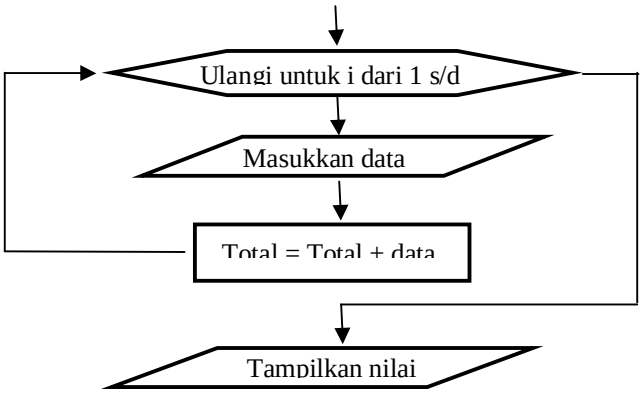
   Penggunaan garis putus-putus sebagai penghubungnya, untuk menpertegas bahwa antara pendeklarasian data dengan instruksi yang ada bukan merupakan urutan statis didalam eksekusi/pengerjaannya. Berbeda dengan garis utuh/*solid* dengan tanda panah (→) yang menunjukkan secara tegas urutan instruksi yang akan dikerjakan.



Gambar 7. Perbaikan Flowchart Menghitung Luas Segitiga

2. Penyesuaian bagan untuk perulangan dengan cacah tertentu (*for-to-do*)

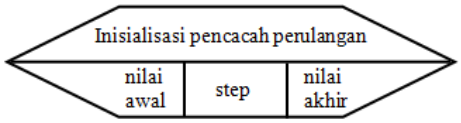
Untuk menyajian langkah yang merupakan perulangan dengan cacah tertentu, telah tersedia bagan berbentuk persegi enam yang digunakan untuk pemberian nilai awal untuk inisialisasi perulangan (*preparation*). Biasanya inisialisasinya disajikan dalam satu instruksi yang di dalamnya tertulis pencacah beserta nilai awal dan nilai akhir perulangan. Contohnya seperti pada Gambar 8, yaitu algoritma untuk menjumlah 10 buah data.



Gambar 8. Blok perulangan dengan cacah tertentu (*for-to-do*)

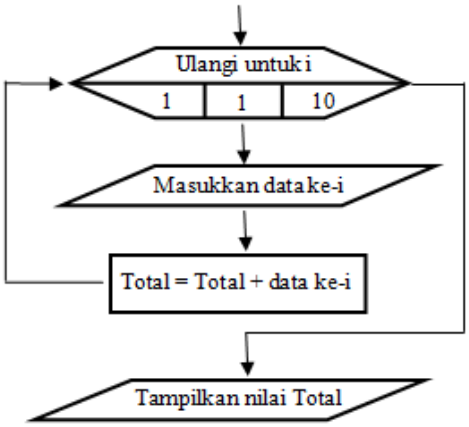
Penyajian algoritma untuk blok perulangan seperti ini (Gambar 8) dapat memunculkan ketidakjelasan bagi mereka yang belum terbiasa dengan konsensus yang ada; Yaitu: kenaikan nilai dari pencacahan jika tidak dituliskan artinya 1, atau penurunannya -1. Pada contoh perulangan ini, nilai pengenal/identifier i sebagai pencacah akan bergerak dari 1, bertambah satu demi satu, sampai berakir pada nilai 10. Sehingga terjadi 10 kali perulangan instruksi yang ada di dalam blok perulangan, memasukkan data dan menjumlahnya.

Bagi mereka yang belum mahir membaca/mmbuat flowchart, notasi perulangan ini dapat memunculkan ketidak jelasan kaitannya dengan kenaikan nilai pencacahnya. Untuk itu perlu dilakukan sedikit perubahan, sehingga bagan segienam terbagi menjadi 4 area seperti pada Gambar 9.



Gambar 9. Bagan perulangan dengan cacah tertentu (*for-to-do*) yang diperbarui

Adanya perbaikan bagan ini menjadi beberapa area menjadikan pengenal dan nilai yang diperlukan memiliki tempatnya masing-masing. Hal ini akan menjadikan proses perulangan yang akan dikenakan kepada blok instruksi yang diulang semakin jelas dan konsisten. Dengan menggunakan bagan yang diperbarui ini, maka algoritma perulangan yang digambarkan sebelumnya dapat disajikan kembali menjadi seperti pada Gambar 10.



Gambar 10. Blok perulangan dengan cacah tertentu (*for-to-do*)
dengan bagan inisialisasi diperbarui

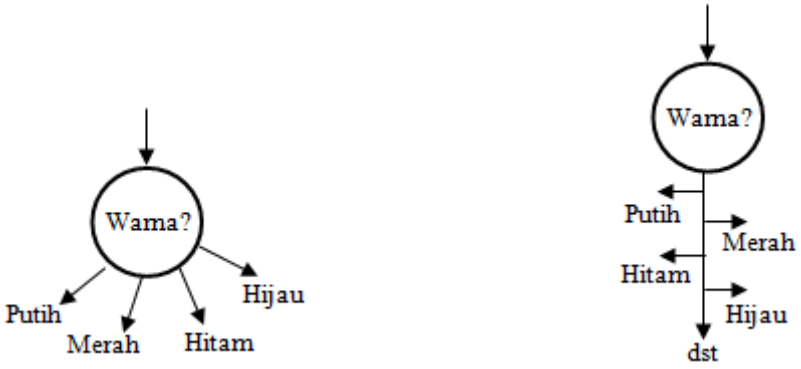
3. Perubahan bagan konektor berupa lingkaran untuk percabangan berdasarkan nilai (*case-of*)

Pengambaran flowchart yang penuh dengan bagan dan garis penghubung sering menyisakan permasalahan kehabisan ruang (satu halaman kertas telah penuh) untuk membuat bagan. Hal ini memaksa pembuatan bagan yang tersisa dituliskan di halaman yang terpisah. Kondisi ini menuntut perlu adanya bagan konektor untuk menghubungkannya. Permasalahan lainnya, banyaknya garis penghubung memungkinkan terjadinya perpotongan garis penghubung satu dengan lainnya di sebuah halaman (kertas) yang sama. Jika ini diabaikan dapat menyebabkan kesalahan penelusuran aliran instruksi yang ada. Untuk itu sering dilakukan pemutusan hubungan dengan diikat bagan konektor. Oleh karena itu, di flowchart pada umumnya disediakan dua bagan konektor, yaitu (a) lingkaran untuk menghubungkan dalam satu halaman yang sama, dan (b) lencana untuk hubungan antara dua halaman yang berbeda.

Walaupun cara ini terlihat tepat, tetapi kurang efesien. Untuk merujuk sambungan dari suatu hubungan di dalam flowchart, baik pada satu halaman yang sama, atau dua halaman yang berbeda, tidak perlu menyediakan dua bagan yang berbeda. Hal ini dikarenakan perujukan sambungan dituliskan beserta bagannya; Sehingga dengan menyertakan nomor halaman dan pengenal/penandanya sudah mewakili kedua jenis koneksi yang ada.

Dari penyederhanaan koneksi ini, maka salah satu bagan konektor, yaitu bagan lingkaran, dialokasikan untuk penyajian instruksi yang lainnya. Alur seleksi yang pemilihannya berdasarkan nilai, bukan kondisional benar atau salah, masih disajikan dengan bagan keputusan (belah ketupat). Hal ini sekilas dapat diasosiasikan, tetapi pada cara pemilihannya jauh berbeda. Cara ini dapat menimbulkan kesalahan konsep antara pemilihan secara langsung berdasarkan nilainya, dengan pemilihan secara berjenjang dengan menggunakan keputusan benar atau salah.

Misalkan akan disajikan alur pemilihan warna yang disukai dengan tersedia 4 pilihan (putih, merah, hitam, dan hijau), maka dengan menggunakan bagan lingkaran dapat disajikan seperti pada Gambar 11.a. atau Gambar 11.b. yang dapat digunakan untuk pemilihan dengan lebih banyak lagi pilihan.



Gambar 11. Blok pemilihan merujuk nilai (*case-of*)
dengan bagan lingkaran

Penggunaan bagan lingkaran untuk alur pemilihan dapat menghindari ketidakkosistenan dengan bagan keputusan yang cara penentuannya berbeda. Selain itu, penulisan instruksi/tindakan untuk setiap pilihan dapat disajikan dengan lebih jelas.

C. SIMPULAN

Dari pembahasan dapat diambil beberapa simpulan terkait penyesuaian cara penyajian algoritma dengan menggunakan flowchart sebagai upaya menjaga konsistensi dan kejelasan langkah penyelesaian masalah dan struktur datanya.

1. Ditambahkan bagan baru berbentuk segi empat dengan sisi-sisi mendatarnya berupa garis ganda digunakan untuk mendeklarasikan pengenal (*identifier*) beserta tipe datanya yang ada/dibutuhkan di dalam proses.
2. Bagan untuk inisialisasi perulangan dengan cacah tertentu (*for-to-do*) diubah dengan penambahan beberapa garis dengan tujuan memperjelas pemanfaatan area pada bagan untuk pengenal/pencacah, nilai awal, nilai kenaikan/penurunan, dan nilai akhir.
3. Bagan lingkaran pemanfaatannya diubah untuk pemilihan yang mengacu pada nilai (*case-of*) agar tidak bias dengan bagan keputusan yang mengacu pada kondisional benar-salah (*if-then-else*).

D. DAFTAR PUSTAKA

Brassard Gilles and Bratley Paul. 1996. *Fundamentals of Algorithmics.* USA: PrenticeHall, Inc.

Gunadarma. (____). *Flowchart*. Diambil tanggal 17 Oktober 2013 dari http://darsono.staff.gunadarma.ac.id/Downloads/files/16512/Flowchart.pdf

Metrowerks CodeWorrior™ CD.1993-1995. *Principles of Programming (ebook)*. Canada and International: Metrowerks. Inc.

Nell Dale and John Lewis. 2011. *Computer Science Illuminated, Fourth Edition*. USA: Jones And Bartlett Publishers