Belajar Memprogram dan Belajar Bahasa Pemrograman
Merupakan Dua Hal yang Berbeda

Oleh : Ema Utami, S.Si, M.Kom

Pada saat kita dihadapkan dengan suatu masalah maka tindakan kita selanjutnya adalah mencari penyelesaiannya. Sekarang kita tidak langsung memecahkan masalah dengan langsung menuliskan solusinya berupa program dalam suatu bahasa pemrograman, namun suatu cara penyelesaian masalah yang akan diprogram dengan menekankan pada desain atau rancangan yang mewakili pemecahan masalah.

Masalah ----------> Algoritma ----------> Penyelesaian
Pemecahan Masalah Tahap Implementasi
(Fase Problem Solving) (Fase Implementasi)

Algoritma mempunyai peranan yang sangat penting dalam bidang teknik informatika pada umumnya dan bidang pemrograman pada khususnya. Algoritma membantu mahasiswa mengembangkan daya penalaran atau kerangka berpikir yang sistematis dalam memahami masalah dan membuat perencanaan atau konsep pemecahan masalah yang lebih baik sehingga dapat menghasilkan yang tepat pula.

Pengertian Algoritma

Kata algoritma, mungkin bukan sesuatu yang asing bagi kita. Penemunya adalah seorang ahli matematika dari uzbekistan yang bernama Abu Abdullah **Muhammad Ibn Musa al-Khwarizmi** (770-840). Di literatur barat dia lebih terkenal dengan sebutan Algorizm. Panggilan inilah yang kemudian dipakai untuk menyebut konsep algorithm yang ditemukannya.

Dalam bahasa Indonesia kita kemudian menyebutkannya sebagai algoritma. Definisi Algoritma adalah logika, metode dan tahapan (urutan) sistematis yang digunakan untuk memecahkan suatu permasalahan.

Dalam beberapa konteks, algoritma adalah spesifikasi urutan langkah untuk melakukan pekerjaan tertentu. Pertimbangan dalam pemilihan algoritma adalah pertama, algoritma haruslah benar. Artinya algoritma akan memberikan keluaran yang dikehendaki dari sejumlah masukan yang diberikan. Tidak peduli sebagus apapun algoritma, kalau memberikan keluaran yang salah, pastilah algoritma tersebut bukanlah algoritma yang baik.

Pertimbangan kedua yang harus diperhatikan adalah kita harus mengetahui seberapa baik hasil yang dicapai oleh algoritma tersebut. Hal ini penting terutama pada algoritma untuk menyelesaikan masalah yang memerlukan aproksimasi hasil (hasil yang hanya berupa pendekatan). Algoritma yang baik harus mampu memberikan hasil yang sedekat mungkin dengan nilai yang sebenarnya.

Ketiga adalah efisiensi algoritma. Efisiensi algoritma dapat ditnjau dari 2 hal yaitu efisiensi waktu dan memori. Meskipun algoritma memberikan keluaran yang benar (paling mendekati), tetapi kalaunkita harus menunggu berjam-jam untuk mendapatkan keluarannya, algoritma tersebut biasanya tidak akan dipakai. Setiap orang menginginkan keluaran yang cepat. Begitu juga dengan memori, semakin besar memori yang terpakai maka semakin buruklah algoritma tersebut.

Dalam kenyatannya, setiap orang bisa membuat algoritma yang berbeda untuk menyelesaikan suatu permasalahan. Walaupun terjadi perbedaan dalam menyusun algoritma, tentunya kita mengharapkan keluaran yang sama. Jika terjadi demikian carilah algoritma yang paling efisien dan cepat.

Beda Algoritma dan Program

Program adalah kompulan instruksi komputer, sedangkan metode dan tahapan sistematis dalam program adalah algoritma. Program ini ditulis dengan menggunakan bahasa pemrograman. Jadi bisa kita sebut bahwa program
adalah suatu implementasi dari bahasa pemrograman.
Beberapa pakar memberi formula bahwa:

Program = Struktur Data + Algoritma

Bagaimanapun juga struktur data dan algoritma berhubungan sangat erat pada sebuah program. Algoritma yang baik tanpa pemilihan struktur data yang tepat akan membuat program menjadi kurang baik, semikian juga sebaliknya.

Pembuatan algoritma mempunyai banyak keuntungan diantaranya:

1. Pembuatan atau penulisan algoritma tidak tergantung pada bahasa pemrograman manapun, artinya penulisan algoritma independen dari bahasa pemrograman dan komputer yang mengeksekusinya.
2. Notasi algoritmik dapat diterjemahkan ke dalam berbagai bahasa pemrograman.
3. Apapun bahasa pemrogramannya, output yang akan dikeluarkan sama karena algoritmanya sama.

Beberapa hal yang perlu dalam membuat algoritma diperhatikan:

1. Teks algoritma berisi deskripsi langkah-langkah penyelesaian masalah. Deskripsi tersebut dapat ditulis dalam notasi apapun asalkan mudah dimengerti dan dipahami.
2. Tidak ada notasi yang baku dalam penulisan teks algoritma seperti pada notasi bahasa pemrograman. Notasi yang digunakan dalam menulis algoritma disebut notasi algoritmik.
3. Tiap orang dapat membuat aturan penulisan dan notasi algoritmik sendiri. Hal ini karena teks algoritma tidak sama dengan teks program. Namun supaya notasi algoritmik mudah ditranslasikan ke dalam notasi bahasa pemrograman tertentu, maka sebaiknya notasi algoritmik tersebut berkorespondensi dengan notasi bahasa pemrograman secara umum.
4. Notasi algoritmik bukan notasi bahasa pemrograman, karena itu pseudocode dalam notasi algoritmik tidak dapat dijalankan oleh komputer. Agar dapat dijalankan oleh komputer, pseudocode dalam notasi algoritmik harus ditranslasikan atau

diterjemahkan ke dalam notasi bahasa pemrograman yang dipilih. Perlu diingat bahwa orang yang menulis program sangat terikat dalam aturan tata bahasanya dan spesifikasi mesin yang menjalankannya.

5.Algoritma sebenarnya digunakan untuk membantu kita dalam mengkonversikan suatu permasalahan ke dalam bahasa pemrograman.

6.Algoritma merupakan hasil pemikiran konseptual, supaya dapat dilaksanakan oleh komputer, algoritma harus ditranslasi ke dalam notasi bahasa pemrograman. Ada beberapa hal yang harus diperhatikan ketika translasi tersebut yaitu:

a)Pendeklarasian variabel.
Apakah bahasa pemrograman yang akan digunakan membutuhkan pendeklarasian variabel karena tidak semua bahasa pemrograman membutuhkannya.

b)Pemilihan tipe data.
Apabila bahasa pemrograman yang akan digunakan membutuhkan pendeklarasian variabel maka perlu dipertimbangkan pada saat pemilihan tipe data.

c)Pemakaian instruksi-instruksi.
Beberapa instruksi mempunyai kegunaan yang sama tetapi masing-masing memiliki kelebihan dan kekurangan yang berbeda.

d)Aturan sintaks.
Pada saat menulis program kita terikat dengan aturan sintaks dari bahasa pemrograman yang akan digunakan.

e)Tampilan hasil.
Pada saat membuat algoritma kita tidak memikirkan tampilan hasil yang akan disajikan. Hal-hal teknis ini kita perhatikan ketika mengkonversikannya menjadi program.

f)Cara pengopersian compiler atau interpreter.
Bahasa pemrograman yang digunakan termasuk kelompok compiler atau interpreter.

Beberapa persyaratan untuk membuat algoritma yang baik adalah:

1.Tingkat kepercayaannya tinggi (realibility). Hasil yang diperoleh dari proses harus berakurasi tinggi dan benar.

2.Pemrosesan yang efisien (cost rendah). Proses harus diselesaikan secepat mungkin dan frekuensi kalkulasi yang sependek mungkin.

3.Sifatnya general. Bukan sesuatu yang hanya untuk menyelesaikan satu kasus saja, tapi juga untuk kasus lain yang lebih general.

4.Bisa dikembangkan (expandable). Haruslah sesuatu yang dapat kita kembangkan lebih jauh berdasarkan perubahan requirement yang ada.

5.Mudah dimengerti. Siapapun yang melihat, dia akan bisa memahami algoritma anda. Susah dimengertinya suatu program akan membuat susah di-maintenance (dikelola).

6.Portabilitas yang tinggi (Portability). Bisa dengan mudah diimplementasikan di berbagai platform komputer.

Penyajian Algoritma

Penyajian algoritma secara garis besar bisa dalam 2 bentuk penyajian yaitu tulisan dan gambar.

Algoritma yang disajikan dengan tulisan yaitu dengan struktur bahasa tertentu (misalnya bahasa Indonesia atau bahasa Inggris) dan pseudocode. Pseudocode adalah kode yang mirip dengan kode pemrograman yang sebenarnya. Pseudocode ditulis berbasis pada bahasa pemrograman tertentu misalnya Pascal, C atau Python, sehingga lebih tepat digunakan untuk menggambarkan algoritma yang akan dikomunikasikan kepada pemrogram. Pseudocode lebih rinci daripada struktur bahasa Inggris, misalnya dalam menyatakan sintaks, tipe data yang digunakan dan lain-lain. Sedangkan algoritma yang disajikan dengan gambar, misalnya dengan flowchart. Flowchart bukan satu-satunya cara untuk menjelaskan atau menerangkan algoritma. Cara yang lain diantaranya :

1.Structure chart
2.DFD (Data Flow Diagram)
3.Warnier diagram
4.IPO (Input Process Output)
5.HIPO (Hierarchical Input Process Output)

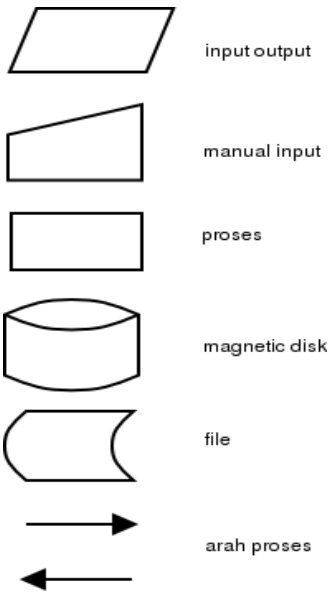
Flowchart (bagan alir) merupakan representasi secara grafik dari suatu algoritma atau prosedur untuk menyelesaikan suatu masalah. Dengan menggunakan flowchart akan memudahkan kita untuk melakukan pengecekan apakah ada bagian-bagian yang terlupakan dalam analisis masalah. Di samping itu flowchart juga berguna sebagai fasilitas untuk berkomunikasi antara pemrogram yang bekerja dalam tim suatu proyek. Flowchart ada dua macam :

1.Flowchart Sistem

Yaitu diagram alir yang menggambarkan suatu sistem peralatan komputer yang digunakan dalam proses pengolahan data dan hubungan antar peralatan tersebut. Flowchart sistem digunakan untuk menggambarkan urutan langkah untuk memecahkan masalah tetapi hanya untuk menggambarkan prosedur dalam sistem yang dibentuk.

Simbol yang digunakan :

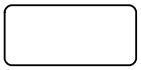
Contoh:



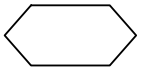
2. Flowchart program

Yaitu bagan yang menggambarkan urutan logika dari suatu prosedur pemecahan masalah.

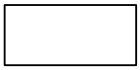
Simbol yang digunakan adalah *American National Standard Inc*.



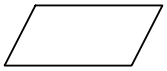
: (*terminal symbol*), menunjukkan awal dan akhir dari program



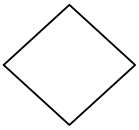
: (*preparation symbol*), memberikan niai awal pada suatu variabel atau *counter*



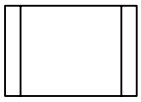
: (*processing symbol*), menunjukkan pengolahan aritmatika dan pemindahan data



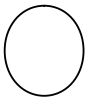
: (*input/output symbol*), menunjukkan proses input atau output



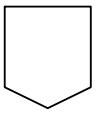
: (*decision symbol*), untuk mewakili operasi perbandingan logika



: (*predefined process symbol*), proses yang ditulis sebagai sub program, yaitu prosedur/ fungsi



: (*connector symbol*), penghubung pada halaman yang sama



: (*off page connector symbol*), penghubung pada halaman yang berbeda

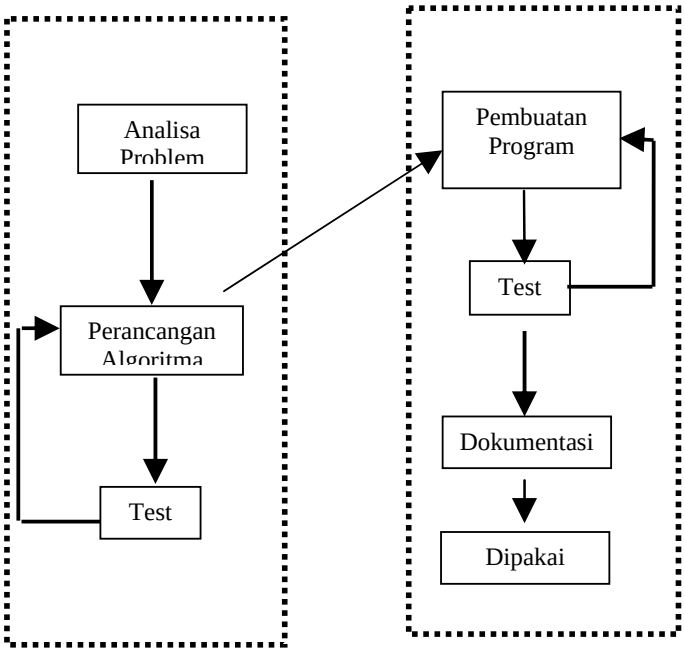
: arah proses

Tahap-Tahap Pemrograman

Sebelumnya perlu dipahami tiga pengertian pokok yakni program, bahasa pemrograman dan pemrograman. Program adalah kata, ekspresi, pernyataan yang disusun dan dirangkai menjadi satu kesatuan prosedur yang berupa urutan langkah untuk menyelesaikan masalah yang diimplementasikan dengan menggunakan bahasa pemrograman sehingga dapat dieksekusi oleh komputer. Bahasa pemrograman adalah

prosedur atau tata cara penulisan program. Sedangkan pemrograman adalah proses mengimplementasikan urutan langkah untuk menyelesaikan suatu masalah dengan menggunakan suatu bahasa pemrograman.

Pemrograman meliputi dua tahapan yaitu:
1.Fase Problem Solving
2.Fase Implementasi



Fase Problem Solving Fase Implementasi

Langkah-langkah untuk dapat menyelesaikan masalah adalah sebagai berikut:
1.Memahami atau menganalisis masalah

Hal-hal yang harus diketahui dalam analisis masalah supaya kita mengetahui bagaimana permasalahan tersebut:

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

2.Merancang atau merumuskan algoritma

Bila masalahnya kompleks maka kita bagi ke dalam modul-modul. Tahap penyusunan algoritma seringkali dimulai dari langkah yang global terlebih dahulu. Langkah global ini diperhalus sampai menjadi langkah yang lebih rinci atau detail. Cara pendekatan ini sangat bermanfaat dalam membuat algoritma untuk masalah yang kompleks. Penghalusan lanngkah dengan cara memecah langkah menjadi beberapa langkah. Tiap langkah diuraikan lagi menjadi beberapa langkah yang lebih sederhana. Penghalusan langkah ini akan terus berlanjut sampai setiap langkah sudah cukup rinci dan tepat untuk dilaksanakan oleh pemroses.

Ciri-ciri algoritma yang baik :

a)Precise (tepat, betul, teliti)

Setiap instruksi harus ditulis dengan seksama dan tidak ada keragu-raguan, dengan demikian setiap instruksi harus dinyatakan secara eksplisit dan tidak ada bagian yang dihilangkan karena pemroses dianggap sudah mengerti. Setiap langkah harus jelas dan pasti.

Contoh: Tambahkan 1 atau 2 pada x.

Instruksi di atas terdapat keraguan.

b)Jumlah langkah atau instruksi berhingga dan tertentu.

Artinya untuk kasus yang sama, banyaknya langkah tetap dan tertentu meskipun datanya berbeda.

c)Efektif

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

Contoh: Hitung akar 2 dengan presisi sempurna.

Instruksi di atas tidak efektif, agar efektif instruksi tersebut diubah. Misal: Hitung akar 2 sampai lima digit di belakang koma.

d)Harus terminate

Jalannya algoritma harus ada kriteria berhenti. Pertanyaannya adalah apakah apabila jumlah instruksinya berhingga maka pasti terminate?

e)Output yang dihasilkan tepat

Jika langkah-langkah algoritmanya logis dan diikuti dengan seksama maka dihasilkan output yang diinginkan.

3.Menulis program

Algoritma yang telah dibuat diterjemahkan dalam bahasa komputer menjadi sebuah program. Ketika menulis program, seorang pemrogram akan terikat dengan sintaks-sintaks instruksi dalam bahasa pemrograman yang digunakan. Hal ini tidak terjadi ketika menyusun atau membuat algoritma karena tidak ada notasi yang baku dalam penulisan teks algoritma seperti pada notasi bahasa pemrograman. Notasi yang digunakan dalam menulis algoritma disebut notasi algoritmik. Tiap orang dapat membuat aturan penulisan dan notasi algoritmik sendiri. Hal ini karena teks algoritma tidak sama dengan teks program. Namun supaya notasi algoritmik mudah ditranslasikan ke dalam notasi bahasa pemrograman tertentu, maka sebaiknya notasi algoritmik tersebut berkorespondensi dengan notasi bahasa pemrograman secara umum.

Notasi algoritmik bukan notasi bahasa pemrograman, karena itu pseudocode dalam notasi algoritmik tidak dapat dijalankan oleh komputer. Agar dapat dijalankan oleh komputer, pseudocode dalam notasi algoritmik harus ditranslasikan atau diterjemahkan ke dalam notasi bahasa pemrograman yang dipilih. Perlu diingat bahwa orang yang menulis program sangat terikat dalam aturan tata bahasanya dan spesifikasi mesin yang menjalankannya. Perlu diperhatikan bahwa pemilihan algoritma yang salah akan menyebabkan program memiki unjuk kerja yang kurang baik.

Program yang baik memiliki standar penilaian :

a. Standar teknik pemecahan masalah

a.Teknik Top-Down

Teknik pemecahan masalah yang paling umum digunakan. Prinsipnya adalah suatu masalah yang kompleks dibagi-bagi ke dalam beberapa kelompok masalah yang lebih kecil. Dari masalah yang kecil tersebut dilakukan analisis. Jika dimungkinkan maka masalah tersebut akan dipilah lagi menjadi subbagian-subbagian dan setelah itu mulai disusun langkah-langkah untuk penyelesaiannya secara lebih detail.

b.Teknik Bottom-Up

Prinsip teknik bottom up adalah pemecahan masalah yang kompleks dilakukan dengan menggabungkan prosedur-prosedur yang ada menjadi satu kesatuan program sebagai penyelesaian masalah tersebut.

b. Standar penyusunan program

a.Kebenaran logika dan penulisan
b.Waktu minimum untuk penulisan program
c.Kecepatan maksimum eksekusi program
d.Ekspresi penggunaan memori
e.Kemudahan merawat dan mengembangkan program
f.User friendly
g.Portability
h.Pemrograman modular

c. Standar perawatan program

a.Dokumentasi
b.Penulisan instruksi

d. Standar prosedur

4.Uji hasil

Pertama kali harus diuji apakah program dapat dijalankan. Apabila program tidak dapat dijalankan maka perlu diperbaiki penulisan sintaksnya tetapi bila program dapat dijalankan maka harus diuji dengan menggunakan data-data yang biasa yaitu data yang diharapkan oleh sistem yang dibuat maupun data-data yang ekstrem yaitu data yang tidak diharapkan oleh sistem. Contoh data ekstrem misalnya program menghendaki masukan jumlah data tetapi user mengisikan dengan bilangan negatif. Program sebaiknya diuji menggunakan data yang relatif banyak. Memperbaiki atau membetulkan suatu error atau kesalahan dapat menjadi sangat mahal karena ada beberapa alasan, diantaranya :

a)Program yang dibuat oleh setiap pemrogram bisa saja strukturnya berbeda walaupun hasilnya sama. Jadi apabila kita tidak bisa memahami maksud programer maka pembetulan program akan memakan waktu yang lama dan tidak efisien.

b)Dalam membetulkan kesalahan program, kita harus melihat kebutuhan sekarang. Jadi bisa saja program yang error tersebut disesuaikan dengan keadaan sekarang.

c)Memperbaiki kesalahan membutuhkan waktu yang lama karena perbaikan terhadap satu error bisa menyebabkan timbulnya error yang lain atau memperbaiki error-error yang lain sehingga semuanya bisa diatasi.

d)Memperbaiki kesalahan tentu saja akan mempelajari maksud program secara keseluruhan beserta struktur datanya. Jadi tentu saja akan membutuhkan banyak waktu, disamping biaya yang dikeluarkan.

5.Membuat dokumentasi

Dokumentasi program ada dua macam yaitu dokumentasi internal dan dokumentasi eksternal. Dokumentasi internal adalah dokumentasi yang dibuat di dalam program yakni setiap kita menuliskan baris program sebaiknya kita beri komentar atau keterangan supaya mempermudah kita untuk mengingat logika yang terdapat dalam instruksi tersebut, hal ini sangat bermanfaat ketika suatu saat program tersebut akan dikembangkan. Dokumentasi eksternal adalah dokumentasi program yang dilakukan dari luar program yaitu membuat user guide atau tbuku petunjuk aturan atau cara menjalankan program tersebut.

6.Program dipakai

Jika program yang telah kita buat sudah sesuai dengan yang kita inginkan maka program terebut dapat kita pakai.

Struktur Dasar Algoritma

Algoritma berisi langkah-langkah penyelesaian suatu masalah. Langkah-langkah tersebut dapat berupa runtunan aksi (sequence), pemilihan aksi (selection), pengulangan aksi (iteration) atau kombinasi dari ketiganya. Jadi struktur dasar pembangun algoritma ada tiga, yaitu :

1.Struktur Runtunan

Digunakan untuk program yang instruksinya sequential atau urutan.

2.Struktur Pemilihan

Digunakan untuk program yang menggunakan pemilihan atau penyeleksian kondisi.

3.Struktur Perulangan

Digunakan untuk program yang instruksinya akan dieksekusi berulang-ulang.

Efisiensi Algoritma

Hal yang perlu diperhatikan dalam pembuatan algoritma adalah waktu proses. Untuk menentukan waktu proses secara tepat merupakan pekerjaan yang sulit. Analisa yang diinginkan untuk menyatakan efisiensi algoritma haruslah dibuat seumum mungkin sehingga bisa dipakai pada semua algoritma, terlepas dari implementasinya. Dalam melakukan analisa suatu algoritma kita ahrus mengfokuskan diri pada operasi aktif yang merupakan pusat program, yaitu bagian program yang paling sering dieksekusi. Bagian-bagian lain seperti pemasukan data, penugasan dan lain sebagainya dapat diabaikan, karena bagian ini tidak dieksekusi sesering operasi aktif. Jumlah keksekusi operasi aktif inilah yang selanjutnya dihitung.

Perhatikan contoh berikut ini:

Jika kita ingin membuat sebuah program untuk menghitung jumlah dari suatu barisan bilangan.

```
Jumlah=0                          (I)

For i=1 to n
  Jumlah=jumlah+i      (II)
End for
```

```
Write(jumlah)                    (III)
```

- Bagian (I) akan dieksekusi sebanyak 1 kali.
- Bagian (II) merupakan sebuah loop (perulangan). Loop ini diproses berdasarkan kenaikan nilai i, dimulai dari i=1 sampai dengan i=n. Jadi dibagian ini yaitu perintah `Jumlah=jumlah+i` akan diproses sebanyak n kali.
- Dan bagian (III) juga akan diproses sebanyak 1 kali.

Bagian (II) merupakan bagian yang paling sering diproses, maka bagian ini merupakan operasi aktif dari algoritma tersebut. Sedangkan bagian (I) dan bagian (II) dapat diabaikan karena bagian-bagian tersebut tidak diproses sesering bagian (II). Waktu proses akan sebanding dengan n. Jadi kalau seandainya kita ingin melakukan efisiensi maka pada bagian (II) inilah yang harus kita pertimbangkan dalam pembuatan algoritmanya.

Resep Pemrograman

Memahami sebuah konsep dasar pembelajaran merupakan modal dasar untuk menguasai materi tersebut. Konsep pemrograman sebenarnya bukanlah menulis source code program, tetapi membuat algoritma dan logika kerja suatu masalah yang akan diselesaikan. Artinya membuat algoritma dan logika kerja suatu masalah merupakan langkah utama yang harus dikerjakan. Setelah selesai baru dikonversikan ke bahasa pemrograman yang diinginkan. Kita tidak harus menguasai seluruh sintaks yang ada dalam bahasa pemrograman yang dipakai. Dan memang sangat sulit untuk menghapal semua sintaks yang ada dalam sebuah bahasa pemrograman.

Apa yang harus kita jika kita tidak mengetahui sintaks-sintaks yang diperlukan? Misalnya kita tidak mengetahui perintah mencetak output dari suatu proses di dalam program yang kita buat. Orang bijak mengatakan kita tidak perlu mengetahui semua hal sebab kita hanyalah manusia biasa dan memang tidak mungkin dilakukan, tetapi yang perlu kita ketahui adalah dimana kita bisa mendapatkan informasi untuk permasalahan tersebut. Banyak sekali buku-buku dan referensi yang dapat kita pakai sebagai acuan pembelajaran. Kesimpulannya bahwa dalam merancang sebuah program yang diutamakan adalah algoritma penyelesaian masalah.

Referensi

1. Utami, E. dan R. Suwanto; *Logika, Algoritma dan Implementasinya dalam Bahasa Python di GNU/Linux*; Penerbit Andi Yogyakarta, 2004
2. Utami, E. dan R. Suwanto; *Belajar C di GNU/Linux*; Graha Ilmu Yogyakarta, 2004
3. Utami, E. dan R. Suwanto; Struktur Data Menggunakan *C di GNU/Linux*; Penerbit Andi Yogyakarta, 2004
4. Utami, E. dan Sukrisno; *10 Langkah Memahami Logika dan Algoritma Menggunakan Bahasa C/C++ di GNU/Linux*; Penerbit Andi Yogyakarta, 2005
5. Utami, E. dan Sukrisno; *101 Tips dan Trik Bahasa C untuk Pemula di GNU/Linux*; Penerbit Andi Yogyakarta, 2005