BAB I

PENGENALAN ALGORITMA

1.1. Pendahuluan

Komputer adalah alat bantu untuk menyelesaikan masalah. Namun, masalah yang ingin diselesaikan tidak dapat langsung "disodorkan" begitu saja ke komputer. Sebagai contoh, andaikan anda mempunyai data nilai ujian sekumpulan mahasiswa. Anda ingin mengurutkan semua data tersebut dari nilai tertinggi hingga nilai terendah. Misalkan anda ketikkan semua nilai ujian dengan menggunakan sebuah program pengolah kata, lalu dibawahnya dituliskan perintah kepada komputer untuk mengurutkan seperti berikut:

30, 27, 36, 91, 64, 76, 74, 32
Urutkan mulai dari nilai terbesar hingga nilai terkecil.

Apakah komputer Anda mengerjakan masalah pengurutan tersebut? Tentu saja tidak, karena komputer tidak "mengerti" perintah anda di atas dan tidak "tahu" bagaimana cara mengurutkan sekumpulan nilai tersebut.

Agar komputer dapat menyelesaikan masalah tersebut, maka Anda perlu merumuskan langkah-langkah penyelesaian tersebut dalam suatu rangkaian instruksi. Komputerlah yang akan mengerjakan rangkaian instruksi tersebut, karena komputer dapat mengerjakannya dengan cepat, akurat, bahkan berulang-ulang tanpa kenal lelah dan bosan. Sekumpulan instruksi yang merupakan penyelesaian masalah itu dinamakan program. Program "dimasukkan" ke dalam komputer, komputer mengerjakan instruksi di dalam program tersebut, lalu memberikan hasil atau keluaran yang diinginkan.
Misalkan program yang berisi rangkaian instruksi untuk mengurutkan sekumpulan data tersebut anda rumuskan kembali sebagai berikut:

Langkah 1 : Cari nilai terbesar diantara N buah data
Langkah 2 : Tempatkan nilai terbesar tersebut pada posisi yang tepat (dengan cara mempertukarkan)
Langkah 3 : Ulangi langkah 1 untuk N – 1 buah data yang lain.

Instruksi di atas masih belum bisa dijalankan oleh komputer karena bahasanya tidak dimengerti oleh komputer. Agar program dapat dilaksanakan oleh komputer, maka program tersebut harus ditulis dalam suatu bahasa yang dimengerti oleh komputer. Sebagaimana dalam kehidupan manusia, kita hanya dapat memberikan perintah kepada orang lain dalam bahasa yang dimengerti olehnya. Karena komputer adalah mesin, maka program harus ditulis dalam bahasa yang khusus dibuat untuk "berkomunikasi" dengan komputer. Bahasa komputer yang digunakan dalam menulis program dinamakan bahasa pemrograman.

Saat ini, dengan berkembangnya teknik pemrograman terstruktur, orang tidak lagi memecahkan masalah dengan langsung menulis programnya dalam bahasa pemrograman. Orang mulai memikirkan suatu cara penyelesaian masalah yang akan diprogram dengan menekankan pada desain atau rancangan yang mewakili pemecahan masalah tersebut. Desain ini independen dari bahasa pemrograman yang digunakan dari komputer yang menjalankan program. Desain menyajikan cara berpikir si pemrogram dalam menyelesaikan masalah. Desain berisi urutan langkah-langkah pencapaian solusi yang ditulis dalam notasi-notasi deskriptif. Urutan langkah-langkah yang sistematis untuk menyelesaikan sebuah masalah dinamakan algoritma.

Notasi yang digunakan untuk menuliskan algoritma disebut notasi algoritma. Notasi algoritma bukan notasi bahasa pemrograman, karena itu program dalam notasi algoritma tidak dapat dijalankan oleh komputer. Agar dapat dijalankan oleh komputer, program dalam notasi algoritma harus ditranslasikan (diterjemahkan) ke dalam notasi bahasa pemrograman yang dipilih.

1.2. Algoritma

Ditinjau dari asal usul kata, kata algoritma sendiri mempunyai sejarah yang tidak lazim. Kata ini tidak muncul di dalam kamus webster sampai akhir tahun 1957. Orang hanya menemukan kata *algorism* yang berarti proses menghitung dengan angka Arab. Anda dikatakan *algorist* jika anda menggunakan angka Arab. Para ahli bahasa berusaha menemukan asal kata algorism ini namu hasilnya kurang memuaskan. Akhirnya para ahli sejarah matematika menemukan asal mula kata tersebut. Kata algorism berasal dari nama penulis buku Arab yang terkenal, yaitu Abu Ja'far Muhammad Ibnu Musa al-Khuwarizmi (al-Khuwarizmi dibaca orang Barat menjadi algorism). Al-Khuwarizmi menulis buku yang berjudul *Kitab al jabar wal-muqabala*, yang artinya "Buku pemugaran dan pengurangan" (*The Book of Restoration and Reduction).* Perubahan kata dari kata *algorism* menjadi *algorithm* muncul karena kata *algorism* sering dikelirukan dengan *arithmetic*, sehingga akhiran –sm berubah menjadi –thm. Karena perhitungan dengan angka Arab sudah menjadi hal yang biasa/lumrah, maka lambat laun kata *algorithm* berangsur-angsur dipakai sebagai metode perhitungan (komputasi) secara umum, sehingga kehilangan makna aslinya. Dalam bahasa Indonesia, kata *algorithm* diserap menjadi algoritma.

Pada tahun 1950, kata algoritma sering dihubungkan dengan 'Algoritma Euclidean' (*Euclid's algorithm*), yaitu proses untuk menemukan pembagi bersama terbesar (*common greatest divisor*) dari dua buah bilangan bulat, m dan n. Misalnya m=80 dan n=12. Faktor pembagi 80 adalah 1,2,4,5,8,10,16,20,40,80 dan faktor pembagi 12 adalah 1,2,3,4,6,12, maka pbt(80,12)=4. Langkah-langkahnya adalah sebagai berikut:

80/12 = 8, sisa 8
12/8 = 1, sisa 4
8/4 = 2 sisa 0

karena pembagian yang terakhir menghasilkan 0, maka sisa pembagian terakhir sebelum 0 yaitu 4, menjadi pbt(80,12). Jadi pbt(80,12) = pbt(12/8) = pbt(8,4) = pbt(4,0) = 4.
Algoritma EUCLIDEAN dapat dituliskan sebagai berikut:

Algoritma EUCLIDEAN
Diberikan dua buah bilangan bulat tak negatif m dan n (m ≥ n). Carilah pembagi bersama terbesar (pbt) dari kedua bilangan tersebut, yaitu bilangan bulat positif terbesar yang habis membagi m dan n.

DESKRIPSI :
1. Jika n = 0 maka
   M adalah jawabannya;
   Stop
   Tetapi jika n ≠ 0,
   Lanjutkan ke langkah 2.
2. Bagilah m dengan n dan misalkan r adalah sisanya.
3. Ganti nilai m dengan nilai n dan nilai n dengan nilai r, lalu ulang kembali ke langkah 1.

Dengan menggunakan m = 80 dan n=12, maka pbt(80,12) dihitung dengan algoritma EUCLIDEAN di atas sebagai berikut:

1. (1) Karena n = 12, maka lanjutkan ke langkah 2
2. (1) Hitung m/n = 80/12 = 6, sisanya r=8.
3. (1) Nilai $m_{baru}=n_{lama}=12$ dan $n_{baru}=r=8$, lanjut ke langkah 1.

1. (2) Karena n = 8, maka lanjutkan ke langkah 2
2. (2) Hitung m/n = 12/8 = 1, sisanya r=4.
3. (2) Nilai $m_{baru}=n_{lama}=8$ dan $n_{baru}=r=4$, lanjut ke langkah 1.

1. (3) Karena n = 4, maka lanjutkan ke langkah 2
2. (3) Hitung m/n = 8/4 = 2, sisanya r=0.
3. (3) Nilai $m_{baru}=n_{lama}=4$ dan $n_{baru}=r=0$, lanjut ke langkah 1.

1. (4) Karena n = 0, maka m = 4 adalah jawabannya. Stop.

Jadi pbt(80,12)=4

Definisi :

- Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis.
- Algoritma adalah urutan logis pengambilan keputusan untuk pemecahan masalah.
- Algoritma adalah urutan langkah-langkah berhingga untuk memecahkan masalah logika atau matematika
- Algoritma adalah logika, metode dan tahapan (urutan) sistematis yang digunakan untuk memecahkan suatu permasalahan.

Kata logis merupakan kata kunci dalam sebuah algoritma. Langkah-langkah di dalam algoritma harus logis, ini berarti hasil dari urutan langkah-langkah tersebut harus dapat ditentukan, benar atau salah. Langkah-langkah yang tidak benar dapat memberikan hasil yang salah.

Sebagai contoh, tinjau persoalan mempertukarkan isi dua buah bejana, A dan B. Bejana A berisi larutan yang berwarna merah dan bejana B berisi larutan berwarna biru. Kita ingin mempertukarkan isi kedua bejana itu sedemikian rupa sehingga bejana A berisi larutan berwarna biru dan bejana B berisi larutan berwarna merah.

Seseorang menuliskan langkah-langkah pertukaran isi kedua bejana sebagai berikut:

Algoritma TUKAR ISI BEJANA 1

Diberikan dua buah bejana, A dan B, bejana A berisi larutan berwarna merah, bejana B berisi larutan berwarna biru. Pertukarkan isi kedua bejana itu sedemikian sehingga bejana A berisi larutan berwarna biru dan bejana B berisi larutan berwarna merah.

DESKRIPSI:
1. Tuangkan larutan dari bejana A ke dalam bejana B.
2. Tuangkan larutan dari bejana B ke dalam bejana A.

Algoritma di atas tidak meghasilkan pertukaran yang benar. Untuk mempertukarkan isi dua buah bejana, kita memerlukan sebuah bejana tambahan yang diperlukan sebagai tempat penampungan sementara. Sebuat bejana tambahan tersebut bejana C. dengan menggunakan bejana bantuan C ini, algoritma mempertukarkan isi dua buah bejana yang benar adalah sebagai berikut:

Algoritma TUKAR ISI BEJANA 2

Diberikan dua buah bejana, A dan B, bejana A berisi larutan berwarna merah, bejana B berisi larutan berwarna biru. Pertukarkan isi kedua bejana itu sedemikian sehingga bejana A berisi larutan berwarna biru dan bejana B berisi larutan berwarna merah.

DESKRIPSI:
1. Tuangkan larutan dari bejana A ke dalam bejana C.
2. Tuangkan larutan dari bejana B ke dalam bejana A.
3. Tuangkan larutan dari bejana C ke dalam bejana B.

Contoh algoritma mempertukarkan isi bejana di atas memberikan dua pesan penting. Pertama, algoritma harus benar. Kedua, algoritma harus berhenti, dan setelah berhenti, algoritma memberikan hasil yang benar.

Menurut Donald E. Knuth dalam bukunya yang berjudul *The Art of Computer Programming*, algoritma harus mempunyai lima ciri penting:

1. Algoritma harus berhenti setelah mengerjakan sejumlah langkah terbatas (berhingga). Barisan instruksi yang dibuat dalam suatu urutan tertentu, dimaksudkan agar masalah yang dihadapi dapat diselesaikan. Banyaknya instruksi atau langkah itu haruslah berhingga. Jika tidak demikian, proses yang dilakukan akan memerlukan waktu yang relatif lebih lama dan diperoleh hasil yang tidak diperlukan atau tidak berhubungan dengan masalah yang ada, bahkan mungkin proses akan terus berlangsung walaupun solusi yang diharapkan telah diperoleh. Hasil akhir yang didapat merupakan solusinya atau informasi tidak ditemukannya solusi. Dengan kata lain, baik dalam kondisi ada solusi ataupun tidak, proses tetap akan berhenti. Sebagai contoh, tinjau kembali algoritma EUCLIDEAN. Pada langkah 1, jika n = 0, algoritma berhenti. Jika n ≠ 0, maka nilai n selalu berkurang sebagai akibat langkah 2 dan 3, dan pada akhirnya nilai n = 0. Program yang tidak pernah berhenti adalah program

yang berisi algoritma yang salah. Suatu prosedur yang hanya akan berhenti jika mempunyai atau menghasilkan solusi disebut semi algoritma.

2. Setiap langkah harus didefinisikan dengan tepat dan tidak berarti-dua (*ambiguous*). Pembaca harus mengerti apa yang dimaksud dengan "m dan n adalah bilangan bulat tak-negatif". Contoh lainnya, pernyataan "bagilah P dengan sejumlah beberapa buah bilang bulat positif" dapat bermakna ganda.
3. Algoritma memiliki nol atau lebih masukan (*input*). Masukan adalah besaran yang diberikan kepada algoritma sebelum algoritma mulai bekerja. Algoritma EUCLIDEAN mempunyai dua buah masukan, m dan n, sedangkan algoritma TUKAR ISI BEJANA memiliki masukan larutan bejana A dan larutan bejana B.
4. Algoritma mempunya nol atau lebih keluaran (output). Keluaran adalah besaran yang memiliki hubungan dengan masukan. Keluaran tersebut tentunya harus merupakan solusi dari masalah yang sedang diselesaikan. Algoritma EUCLIDEAN mempunyai satu keluaran yaitu n pada langkah 2 yang merupakan pembagi bersama terbesar dari kedua masukannya. Algoritma TUKAR ISI BEJANA tidak memiliki keluaran sama sekali.
5. Algoritma harus efektif dan efisien. Setiap langkah harus sederhana sehingga dapat dikerjakan dalam sejumlah waktu yang masuk akal. Suatu algoritma dikatakan efektif jika algoritma tersebut dapat menghasilkan suatu solusi yang sesuai dengan masalah yang diselesaikan. Dengan kata lain suatu algoritma harus tepat guna. Suatu algoritma dikatakan efisien jika waktu proses dari algoritma relatif lebih singkat dan penggunaan memorinya lebih sedikit.

> Masalah disebut dapat diselesaikan secara algoritma jika dapat ditulis program komputer yang dapat menghasilkan jawaban benar untuk sembarang masukan (yang dispesifikasikan) dalam waktu yang berhingga dan menggunakan ruang memori yang tersedia.

Pada beberapa sumber lain, ada tambahan ciri dari algoritma:

6. Algoritma harus terstruktur. Urutan baris langkah-langkahnya yang digunakan harus disusun sedemikian rupa agar proses penyelesaiannya tidak berbelit-belit, sehingga memungkinkan waktu prosesnya akan menjadi relatif lebih singkat. Hal ini akan memperlihatkan bahwa bagian-bagian dari proses tersebut dapat dibedakan secara jelas (bagian input, proses dan output). Dengan demikian memudahkan kita didalam melakukan pemeriksaan ulang.

> Suatu algoritma harus menghasilkan output yang tepat guna (efektif) dalam waktu yang realtif singkat dan penggunaan memori yang relatif sedikit (efisien) dengan langkah yang berhingga dan prosesnya berakhir baik dalam keadaan diperoleh suatu solusi maupun tidak adanya solusi.

Algoritma Merupakan Jantung Ilmu Informatika

Algoritma adalah jantung ilmu komputer atau informatika. Banyak cabang ilmu komputer yang di acu dalam terminologi algoritma. Namun, jangan beranggapan algoritma selalu identik dengan ilmu komputer saja. Dalam kehidupan sehari-hari pun banyak terdapat proses yang dinyatakan dalam suatu algoritma. Berikut adalah beberapa contoh algoritma dalam kehidupan sehari-hari.

Tabel 1.1. Contoh Algoritma dalam Kehidupan Sehari-hari

NO	PROSES	ALGORITMA	CONTOH LANGKAH DALAM ALGORITMA
1	Membuat kue	Resep kue	Masukkan telur ke dalam wajan, kocok sampai mengembang.
2	Membuat pakaian	Pola pakaian	Gunting kain dari pinggir kiri bawah ke arah kana sejauh 5 cm
3	Praktikum reaksi kimia	Panduan praktikum	Campurkan 10 ml H_2SO_4 ke dalam 15 ml NaOH
4	Merakit mobil	Panduan merakit	Sambungkan komponen A dengan komponen B
5	Kegiatan sehari-hari	Jadwal harian	Pukul 15 : tidur siang Pukul 16 : membuat PR
6	Memainkan musik	Papan not balok	Not balok
7	Mengisi voucher HP	Panduan pengisian	Tekan nomor 888 Masukkan kode voucher

Langkah-langkah pada algoritma haruslah logis. Secara umum, pihak (benda) yang mengerjakan proses disebut pemroses (*processor*). Pemroses tersebut dapat berupa manusia, komputer, robot, atau alat-alat mekanik/elektronik lainnya. Pemroses melakukan suatu proses dengan melaksanakan atau mengeksekusi algoritma yang menjabarkan proses tersebut. Melaksanakan algoritma berarti mengerjakan langkah-langkah di dalam algoritma tersebut.

Pemroses mengerjakan proses sesuai dengan algoritma yang diberikan kepadanya. Karena itu suatu algoritma harus dinyatakan dalam bentuk yang dapat dimengerti oleh pemroses. Jadi suatu pemroses harus:

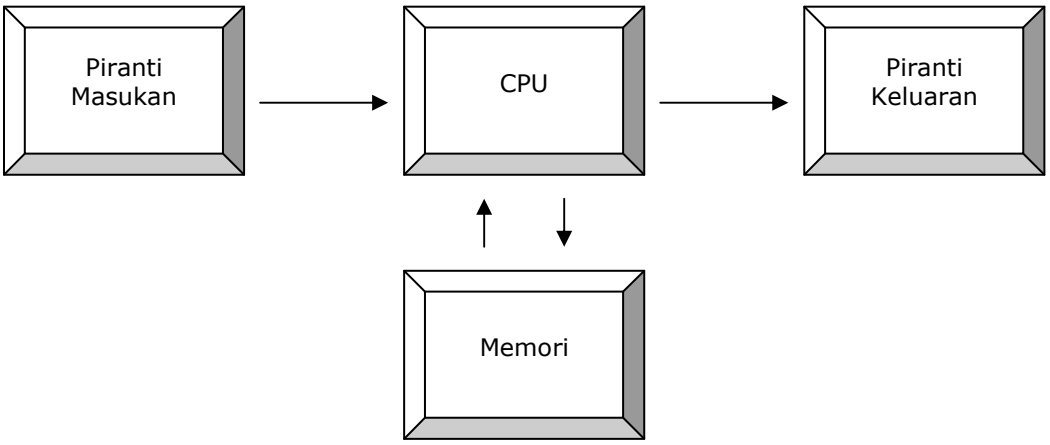
1. Mengerti setiap langkah dalam algoritma.
2. Mengerjakan operasi yang bersesuaian dengan langkah tersebut.

Mekanisme Pelaksanaan Algoritma oleh Pemroses

Komputer hanyalah salah satu pemroses. Agar dapat dilaksanakan oleh komputer, algoritma harus ditulis dalam notasi bahasa pemrograman sehingga dinamakan program.jadi, program adalah perwujudan atau implementasi teknis algoritma yang ditulis dalam bahasa pemrograman tertentu sehingga dapat dilaksanakan oleh komputer.

Kata algoritma dan kata program seringkali dipertukarkan dalam penggunaannya. Algoritma adalah urutan langkah-langkah penyelesaian masalah sedangkan program adalah realisasi algoritma dalam bahasa pemrograman. Program ditulis dalam salah satu bahasa pemrograman dan kegiatan membuat program disebut pemrograman (*programming*). Orang yang menulis program disebut pemrogram (*programmer*). Tiap-tiap langkah di dalam program disebut pernyataan atau instruksi. Jadi, program tersusun atas sederetan instruksi. Bila suatu instruksi dilaksanakan, maka operasi-operasi yang bersesuaian dengan instruksi tersebut dikerjakan oleh komputer.

Secara garis besar komputer tersusun atas empat komponen utama, piranti masukan, piranti keluaran, unit pemroses utama dan memori. Unit pemroses utama (Central Processing Unit – CPU) adalah "otak" komputer, yang berfungsi mengerjakan operasi-operasi dasar (perbandingan, perhitungan, membaca, menulis). Memori adalah komponen yang berfungsi untuk menyimpan program (berisi operasi-operasi yang akan dikerjakan oleh CPU) dan data atau informasi (sesuatu yang diolah oleh operasi-operasi). Piranti masukan dan keluaran (I/O devices) adalah alat yang memasukkan data atu program ke dalam memori, dan alat yang digunakan komputer untuk mengkomunikasikan hasil-hasil aktivitasnya.



Gambar 1.1. Mekanisme Kerja Komponen Utama Komputer

Mekanisme kerja keempat komponen di atas dapat dijelaskan sebagai berikut. Mula-mula program dimasukkan ke dalam memori komputer. Ketika program dilaksanakan (*execute*), setiap instruksi yang telah tersimpan di dalam memori dikirim ke CPU. CPU mengerjakan operasi-operasi yang bersesuaian dengan instruksi tersebut. Bila suatu operasi memerlukan data, data dibaca dari piranti masukan, disimpan di dalam memori lalu dikirim ke CPU untuk operasi yang memerlukannya tadi. Bila proses menghasilkan keluaran atau informasi, keluaran disimpan ke dalam memori, lalu memori menuliskan keluaran tadi ke piranti keluaran (misalkan dengan mencetaknya ke layar peraga).

Belajar Memprogram dan Belajar Bahasa Pemrograman

Belajar memprogram tidak sama dengan belajar bahasa pemrograman. Belajar memprogram adalah belajar tentang metodologi pemecahan masalah, kemudian menuangkannya ke dalam notasi tertentu yang mudah dibaca dan dipahami, sedangkan belajar bahasa pemrograman berarti belajar memakai suatu bahasa, aturan tata bahasanya, instruksi-instruksinya, tata cara pengoperasian *compiler*-nya, dan memanfaatkan instruksi-instruksi tersebut untuk membuat program yang ditulis hanya dalam bahasa itu saja. Belajar memprogram bersifat pemahaman persoalan, analisis dan sintesis dan dititikberatkan pada desain program, sedangkan belajar bahasa pemrograman dititikberatkan pada *coder*.

Berdasarkan terapannya, bahasa pemrograman dapat digolongkan atas dua kelompok besar:

1. **Bahasa pemrograman bertujuan khusus**, yang termasuk kelompok ini adalah COBOL (terapan bisnis dan administrasi), Fortran (terapan komputasi ilmiah), bahasa rakitan (terapan pemrograman mesin), prolog (terapan kecerdasan buatan), dan lain-lain.
2. **Bahasa pemrograman bertujuan umum**, yang dapat digunakan untuk berbagai aplikasi. Yang termasuk kelompok ini adalah bahasa Pascal, Basic dan C.

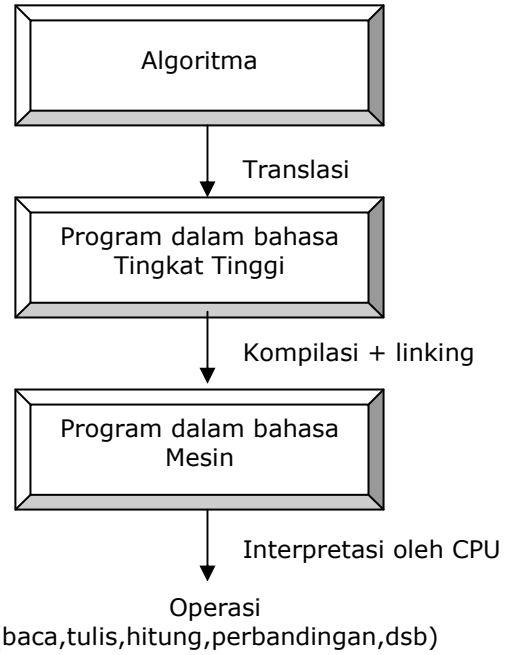
Tentu saja pembagian ini tidak kaku. Bahasa-bahasa bertujuan khusus tidak berarti tidak bisa digunakan untuk aplikasi lain. Cobol misalnya, dapat juga digunakan untuk terapan ilmiah, hanya saja kemampuannya terbatas. Yang jelas, bahasa-bahasa pemrograman yang berbeda dikembangkan untuk bermacam-macam terapan yang berbeda pula.

Berdasarkan pada apakah notasi bahasa pemrograman lebih dekat ke mesin atau ke bahasa manusia, maka pemrograman dikelompokkan atas dua bagian:

1. **Bahasa tingkat rendah**. Bahasa jenis ini dirancang agar setiap instruksinya langsung dikerjakan oleh komputer, tanpa harus melalui penerjemah (*translator*). Contohnya adalah bahasa mesin. CPU mengambil instruksi dari memori, langsung mengerti dan langsung mengerjakan operasinya. Bahasa tingkat rendah bersifat primitif, sangat sederhana, orientasinya lebih dekat ke mesin, dan sulit dipahami manusia. Bahasa rakitan dimasukkan ke dalam kelompok ini karena alasan notasi

yang dipakai dalam bahasa ini lebih dekat ke mesin, meskipun untuk melaksanakan instruksinya masih diperlukan penerjemahan ke dalam bahasa mesin.

2. **Bahasa tingkat tinggi**, yang membuat pemrograman menjadi lebih mudah untuk dipahami, lebih manusiawi dan berorientasi ke bahasa manusia (bahasa inggris). Hanya saja, program dalam bahasa tingkat tinggi tidak dapat langsung dilaksanakan oleh komputer. Ia perlu diterjemahkan terlebih dahulu oleh sebuah *translator* bahasa (*compiler* atau *interpreter*) ke dalam bahasa mesin sebelum akhirnya dieksekusi oleh CPU.



Gambar 1.2. Tahapan pelaksanaan program oleh komputer

Algoritma ditranslasikan menjadi program dalam bahasa tingkat tinggi. Selanjutnya, program dikompilasi dan diterjemahkan menjadi program dalam bahasa mesin dan di-link dengan berkas library. Instruksi dalam bahasa mesin diinterpretasi oleh CPU. Operasi yang bersesuaian dengan setiap instruksi dilaksanakan.

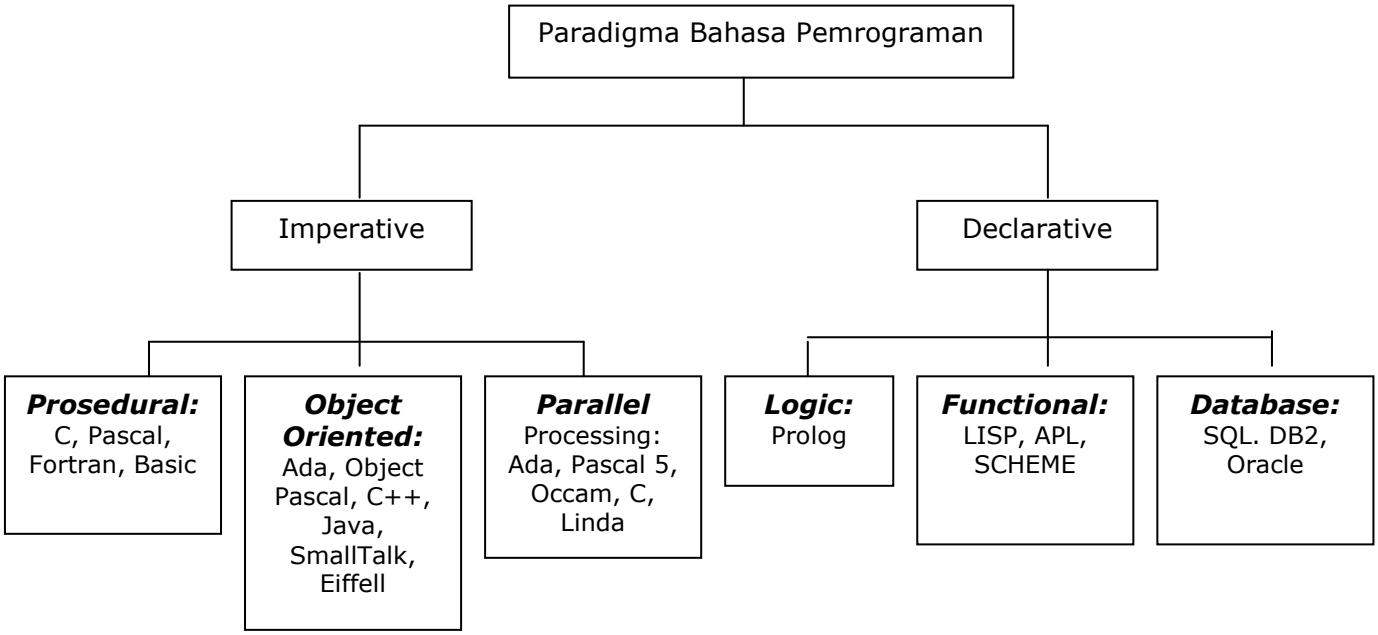
Penerjemah terdiri dari dua jenis, interpreter dan compiler. Perbedaan antara interpreter dan compiler dapat dilihat pada tabel 1.2.

Tabel 1.2. Perbedaan Interpreter dan Compiler

	INTERPRETER	**COMPILER**
1	Menerjemahkan instruksi per instruksi	Menerjemahkan secara keseluruhan
2	Source program tidak harus ditulis lengkap	Source program harus ditulis lengkap
3	Bila terjadi kesalahan kompilasi, dapat langsung dibenarkan secara interaktif	Bila terjadi kesalahan dalam kompilasi, source program harus dibenarkan dan proses kompilasi diulang kembali
4	Tidak menghasilkan objek program	Menghasilkan objek program
5	Tidak menghasilkan executable program karena langsung dijalankan pada saat program diinterpretasi.	Menghasilkan executable program, sehingga dapat dijalankan di keadaan prompt system.
6	Proses interpretasi terasa cepat, karena setiap instruksi langsung dikerjakan dan dapat dilihat hasilnya	Proses kompilasi terasa lama, karena sekaligus menerjemahkan seluruh instruksi program.
7	Source program terus dipergunakan karena tidak dihasilkan executable program	Source program sudah tidak dipergunakan lagi untuk mengerjakan program.
8	Proses pengerjaan program lebih lambat, karena setiap instruksi yang dikerjakan harus diinterpretasi ulang	Proses pengerjaan program lebih cepat, karena executable program sudah dalam bahasa mesin

9	Keamanan program kurang terjamin, karena selalu menggunakan source program	Keamanan program lebih terjamin, karena yang digunakan executable program.

Produk yang dihasilkan program:
- Program dengan rancangan yang baik (metodologis, sistematis)
- Dapat dieksekusi oleh mesin
- Berfungsi dengan benar
- Sanggup melayani segala kemungkinan masukan
- Disertai dokumentasi



Gambar 1.3.Paradigma Bahasa Pemrograman

Tabel 1.2. Bahasa Pemrograman untuk tujuan tertentu.

Jenis Program	Bahasa Terbaik	Bahasa Terburuk
Data terstruktur	ADA, C /C++, PASCAL	Assembler, BASIC
Proyek cepat	BASIC	PASCAL, ADA, Assembler
Eksekusi cepat	Assembler, C	BASIC, Intrepreter Language
Kalkulasi matematika	FORTRAN	PASCAL
Menggunakan memori dinamis	PASCAL, C	BASIC
Lingkungan bermemori terbatas	BASIC, Assembler, C	FORTRAN
Program real-time	ADA, Assembler, C	BASIC, FORTRAN
Manipulasi string	BASIC, PASCAL	C
Program mudah dikelola	PASCAL, ADA	C, FORTRAN

Untuk menyusun sebuah program yang besar dan kompleks, pemrogram membutuhkan tahapan penyusunan yang sistematis dan terpadu, yaitu:

1. Definisi Masalah
2. Analisis Kebutuhan
3. Penyusunan Algoritma
4. Pengkodean/Pemrograman
5. Testing dan Debugging
6. Pemeliharaan
7. Dokumentasi

Penyajian Algoritma

Notasi algoritma dapat diterjemahkan ke dalam berbagai bahasa pemrograman. Notasi algoritma bukan notasi bahasa pemrograman, sehingga siapa pun dapat mebuat notasi algoritma yang berbeda. Hal yang penting mengenai notasi tersebut adalah mudah dibaca dan dimengerti. Selain itu, meskipun notasi algoritma bukan notasi baku sebagaimana pada notasi bahasa pemrograman, ketaatasasan terhadap notasi perlu diperhatikan untuk menghindari kekeliruan.

Teknik penyajian algoritma secara garis besar dibagi menjadi dua yaitu teknik tulisan dan gambar.

A. Teknik Tulisan

- **English Structure**

  English structure menyatakan langkah-langkah algoritma yang akan dikomunikasikan kepada pemakai sistem dengan untaian kalimat deskriptif menggunakan bahasa manusia, biasanya menggunakan bahasa inggris.

  Contoh :
  Algoritma EUCLIDEAN
  Diberikan dua buah bilangan bulat tak negatif m dan n (m ≥ n). carilah pembagi bersama terbesar (pbt) dari kedua bilangan tersebut, yaitu bilangan bulat positif terbesar yang habis membagi m dan n.
  DESKRIPSI :
  1. Jika n = 0 maka
     M adalah jawabannya;
     Stop
     Tetapi jika n ≠ 0,
     Lanjutkan ke langkah 2.
  2. Bagilah m dengan n dan misalkan r adalah sisanya.
  3. Ganti nilai m dengan nilai n dan nilai n dengan nilai r, lalu ulang kembali ke langkah 1.

- **Pseudocode**

  Pseudocode adalah notasi yang menyerupai notasi bahasa pemrograman tingkat tinggi. Keuntungan menggunakan notasi Pseudocode adalah kemudahan mengkonversinya lebih tepat disebut mentranslasi ke notasi bahasa pemrograman,

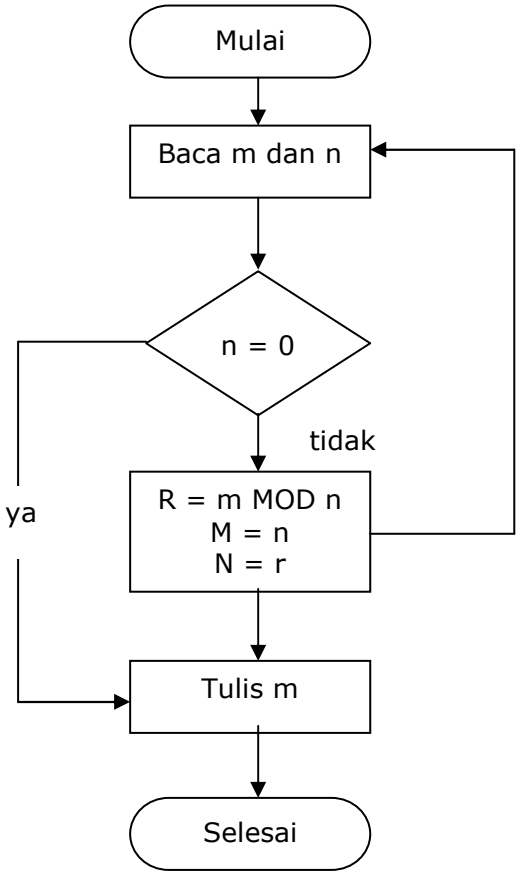
karena terdapat korespondensi antar setiap pseudocode dengan notasi bahasa pemrograman.

Contoh:

```
Algoritma EUCLIDEAN
{Dibaca dua buah bilangan bulat tak negatif m dan n (m ≥ n). Carilah pembagi bersama
terbesar (pbt) dari kedua bilangan tersebut, yaitu bilangan bulat positif terbesar yang habis
membagi m dan n.}
DEKLARASI:
  m, n    : integer       { bilangan bulat yang akan dicari pbt-nya )
  r       : integer       { sisa hasil bagi }
DESKRIPSI:
  read(m,n)       {m ≥ n }
  while n ≠ 0 do
          r ← m MOD n   {hitung sisa hasil pembagian }
          m ← n
          n ← r
  endwhile
  { kondisi selesai pengulangan: n = 0, maka pbt(m,n) = m }
  write (m)
```

B. Teknik Gambar

- Flowchart



Gambar 1.4. Flowchart Algoritma Euclidean

- Structure Chart, digunakan untuk mendefinisikan dan mengilustrasikan organisasi dari sistem secara berjenjang, berbentuk modul dan submodul, menunjukkan hubungan elemen data dan elemen kontrol seta hubungan antar modul.
- Hierarchy plus input-process-output
- Nassi Schneiderman chart

Aturan Penulisan Teks Algoritma

Pada dasarnya teks algoritma selalu disusun oleh tiga bagian (blok) yaitu bagian judul (*header*), bagian deklarasi dan bagian deskripsi algoritma. Setiap bagian disertai dengan komentar untuk memperjelas maksud teks yang dituliskan. Komentar adalah kalimat yang diapit oleh pasangan tanda kurung kurawal.

Algoritma NAMA_ALGORITMA
{ *Penjelasan tentang algoritma, yang berisi uraian singkat mengenai apa yang dilakukan oleh algoritma* }

DEKLARASI
{ *Semua nama yang dipakai, meliputi nama tipe, nama tetapan, nama peubah, nama prosedur, dan nama fungsi didefinisikan disini* }

DESKRIPSI
{ *Semua langkah/aksi algoritma dituliskan disini* }

Dalam memberikan nama algoritma harus mempunyai makna yang mencerminkan proses, sifat atau identitas lainnya yang melekat dengan suatu proses, tipe, konstanta, variabel, sub-program dan lain-lainnya. Nama-nama yang bermakna disebut mnemonic.

Contoh:
Algoritma LUAS_LINGKARAN
{*Menghitung luas lingkaran dengan ukuran jari-jari tertentu .Algoritma menerima masukan jejari lingkaran, menghitung luasnya, dan menyajikan hasilnya ke piranti keluaran*}

DEKLARASI
{*nama konstanta*}
const PHI = 3.14; { Nilai phi = 22/7 }
{*nama peubah*}
var R : real; {*input jejari lingkaran bilangan riil*}
l_Lingkaran : real; {*luas lingkaran bilangan riil*}
{ *nama sub program* }
procedure TUKAR (input/output A:integer, input/output B:integer)
{*Mempertukarkan nilai A dan B.Parameter A dan B sudah terdefinisi nilainya. Setelah pertukaran, A berisi nilai B dan B berisi nilai A*}

DESKRIPSI
{*Baca data jejari lingkaran R.Jika R <= 0 tulis pesan data salah, selain itu hitung luas ingkaran. Tampilkan luas lingkaran*}
baca(R);
jika R <= 0 then tulis ("Data salah !") selain itu l_Lingkaran = PHI x R x R;
tulis(l_Lingkaran);