

PENDAHULUAN



Overview

Komputer merupakan istilah yang umum yang sudah dikenal oleh masyarakat banyak pada umumnya. Komputer sudah menjadi alat bantu kehidupan manusia sehari-hari. Tanpa bantuan manusia, komputer hanya akan menjadi seonggok mesin yang tidak bisa melakukan apa-apa. Program menjadi “roh” yang dapat membuat komputer dapat bekerja dan memberi bantuan kepada manusia. Dalam membuat program harus melalui beberapa tahapan, salah satunya adalah tahap desain. Supaya perancangan program dapat dikomunikasikan dengan orang lain maka, perancangan program harus menggunakan notasi yang standar dan mudah untuk dibaca dan dipahami. Dengan kata lain komputer dan program merupakan satu kesatuan yang saling berkaitan satu sama lain.



Tujuan

- ☑ Memahami bagaimana komputer menangani data elektronik
- ☑ Memahami komponen yang terlibat dalam memproduksi informasi
- ☑ Memahami perbedaan bahasa pemrograman di setiap tingkatan

1.1 Komputer Elektronik

Komputer di era modern seperti sekarang ini, sudah menjadi kebutuhan untuk mendukung aktivitas yang dilakukan oleh manusia. Bentuk fisik dari komputer pun juga beragam, kompak dan semakin praktis.

Seluruh perangkat elektronik pada umumnya terdapat sebuah komputer kecil yang berfungsi sebagai ‘otak’ atau pusat pengendali perangkat tersebut.

Perangkat komputer modern dapat bekerja apabila terdapat energi listrik, demikian pula dengan data yang diolah. Dengan ditemukannya energi listrik, seluruh data dalam bentuk apapun sangat dimungkinkan untuk direpresentasikan ke dalam bentuk elektronik.



Gambar 1.1 Komputer elektronik

1.2 Komponen Komputer

Di dalam sebuah komputer elektronik terdapat beberapa komponen/perangkat yang berfungsi untuk mengolah data. Secara umum, komponen komputer terbagi menjadi 3 (tiga) bagian, yaitu:



Alat input berfungsi sebagai media untuk memasukkan data ke dalam komputer. Contoh alat input adalah: keyboard, mouse, microphone, dll.

Alat pemroses di dalam komputer berfungsi untuk melakukan pengolahan data menjadi informasi. Contoh alat pemroses adalah: prosesor.

Alat output berfungsi sebagai media untuk menyampaikan informasi hasil pengolahan, bisa dalam bentuk tampilan menggunakan monitor ataupun dalam bentuk cetakan menggunakan printer.

Sesungguhnya, komputer itu hanyalah mesin elektronik yang tersusun atas komponen-komponen di atas. Namun dengan adanya energi listrik dan perangkat lunak, barulah komponen komputer dapat aktif dan kemudian digunakan untuk bekerja.

1.3 Algoritma

Kata '*algoritma*' diturunkan dari nama belakang seorang tokoh matematikawan Persia bernama Muhammad ibn Musa al-Khuwarizmi (lahir tahun 730an, meninggal antara tahun 835 dan 850). Al-Khuwarizmi berasal dari propinsi Khorasan di negara yang saat ini bernama Uzbekistan. Uni Soviet menghormati jasa-jasa Al-Khuwarizmi dengan membuat gambar dirinya sebagai perangk.

Algoritma merupakan metode umum yang digunakan untuk menyelesaikan kasus-kasus tertentu. Dalam menuliskan algoritma, dapat digunakan bahasa natural atau menggunakan notasi matematika, sehingga masih belum dapat dijalankan pada komputer.

Dalam kehidupan sehari-hari, kita sudah melakukan penyusunan algoritma untuk menyelesaikan permasalahan atau tantangan yang dihadapi. Sebagai contoh, pada saat diminta untuk membuat telur dadar. Sebelum membuat algoritmanya, kita perlu mendefinisikan masukan (input) dan keluaran (output) terlebih dahulu, dimana input berupa telur mentah, dan output berupa telur dadar yang sudah matang. Susunan algoritmanya sebagai berikut:

- Nyalakan api kompor
- Tuangkan minyak ke dalam wajan
- Pecahkan telur ayam ke dalam mangkok
- Tambahkan garam secukupnya
- Aduk campuran telur dan garam
- Tuang adonan telur ke dalam wajan
- Masak telur hingga matang

Algoritma akan lebih baik jika ditulis secara sistematis menggunakan beberapa skema, dalam buku ini akan dibahas mengenai skema *Flowchart* dan *Pseudocode*.

1.4 Program

Program adalah formulasi sebuah algoritma dalam bentuk bahasa pemrograman[1], sehingga siap untuk dijalankan pada mesin komputer. Membuat program seperti memberitahukan apa yang harus dilakukan kepada orang lain. Sebagai contoh, pada saat kita memberitahukan algoritma membuat telur dadar kepada orang lain, kita sudah melakukan pemrograman.

Pemrograman membuat telur dadar kepada orang lain akan lebih mudah karena orang tersebut sudah mengetahui apa itu telur dadar. Pada langkah yang ke-3 diminta untuk memecahkan telur, bagaimana cara orang tersebut memecahkan telur tentunya sudah diketahui dan kita tidak perlu menjelaskan terlalu detail.

Lain halnya jika kita harus menyuruh komputer untuk melakukan apa yang kita inginkan. Komputer sebenarnya hanyalah sebuah mesin bodoh yang tidak memiliki emosi dan kemampuan bersosialisasi. Oleh karena itu, untuk membuatnya menjadi mudah, diperlukan penyusunan algoritma yang benar.

Mendesain algoritma yang benar dan menterjemahkannya ke dalam bahasa pemrograman bukanlah hal yang mudah karena bahasa pemrograman memiliki tata penulisan sendiri.

1.5 Bahasa Pemrograman

Bahasa pemrograman adalah bahasa buatan yang digunakan untuk mengendalikan perilaku dari sebuah mesin, biasanya berupa mesin komputer, sehingga dapat digunakan untuk memberitahu komputer tentang apa yang harus dilakukan.

Struktur bahasa ini memiliki kemiripan dengan bahasa natural manusia, karena juga tersusun dari elemen-elemen dasar seperti: kata benda dan kata kerja serta mengikuti aturan untuk menyusunnya menjadi kalimat.

1.5.1 Klasifikasi Menurut Generasi

↳ First Generation Language (1GL)

Bahasa pemrograman ini berupa kode-kode mesin yang hanya bisa dipahami oleh mikroprosesor.

↳ Second Generation Language (2GL)

Bahasa pada generasi ini adalah *assembly language*, dimana bahasa ini masih menggunakan kode-kode yang disebut dengan *mnemonic*. Bahasa *assembly* disebut sebagai generasi kedua karena bahasa ini bukan bahasa asli mikroprosesor, meskipun begitu programmer tetap harus mengetahui keunikan dari masing-masing mikroprosesor (register dan jenis instruksi).

↳ Generasi ketiga

Bahasa pemrograman generasi ketiga sengaja didesain supaya mudah dipahami oleh manusia. Pada generasi ini mulai dikenalkan istilah variabel, tipe data, ekspresi aljabar dan sudah mendukung pemrograman terstruktur.

Contoh bahasa: FORTRAN, COBOL, ALGOL, BASIC, C, C++, Pascal, Java.

↳ Generasi keempat

Pada generasi ini, bahasa pemrograman didesain untuk mengurangi *effort* dan mempercepat proses pembuatan program. Pada 3GL, pembuatan program membutuhkan waktu yang lama dan mudah sekali didapati error. Pada 4GL, telah menggunakan metodologi dimana sebuah perintah dapat menghasilkan beberapa instruksi 3GL yang kompleks dengan sedikit error. Contoh bahasa:

- ◆ Pemrograman umum : DataFlex, WinDev, PowerBuilder
- ◆ Basis data : SQL, Progress 4GL
- ◆ Manipulasi data, analisis dan pelaporan : ABAP, Matlab, PL/SQL.

↳ Generasi kelima

Bahasa pemrograman generasi kelima disebut sebagai *constraint-programming* atau *declarative-programming*. Program tidak dituliskan dalam bentuk algoritma melainkan dituliskan batasan atau fakta dari sebuah lingkup masalah, sehingga program akan menghasilkan luaran dalam bentuk

solusi. Bahasa pemrograman ini digunakan untuk membangun sistem kecerdasan buatan dan belum digunakan secara meluas di dunia industri. Contoh bahasa: Prolog, LISP, Mercury.

1.5.2 Klasifikasi Menurut Tingkatan

↳ Low-level programming language

Tingkat bahasa pemrograman ini disebut "rendah" (*low level*) bukan karena posisinya berada di bawah, melainkan karena kurangnya abstraksi (penggambaran kode instruksi) antara bahasa natural dengan bahasa mesin. Oleh karena itu, bahasa di tingkat ini sering disebut sebagai 'bahasa mesin'.

Bahasa pemrograman yang masuk kategori ini adalah bahasa mesin itu sendiri (1GL) dan bahasa *assembly* (2GL).

↳ High-level programming language (HLL)

Bahasa pemrograman di tingkat ini memiliki abstraksi yang lebih banyak dan terdapat kemiripan dengan bahasa natural (bahasa Inggris), lebih mudah untuk digunakan dan mudah untuk dipindahkan antar platform.

↳ Very high-level programming language (VHLL)

Bahasa ini memiliki abstraksi yang lebih tinggi dibandingkan HLL, dan digunakan untuk menunjang produktifitas programmer profesional. Biasanya VHLL digunakan hanya untuk tujuan yang spesifik, misalnya untuk keperluan bisnis: mengolah data, membuat laporan, dsb.

1.6 Paradigma Pemrograman

Paradigma pemrograman merupakan sebuah cara pandang seorang programmer dalam menyelesaikan sebuah masalah dan memformulasikannya kedalam sebuah bahasa pemrograman. Terdapat beberapa paradigma pemrograman, antara lain:

1.6.1 Paradigma Imperatif

Inti dari paradigma ini adalah menjalankan sebuah urutan perintah, jalankan satu perintah kemudian jalankan perintah yang selanjutnya. Sebuah program imperatif tersusun dari sekumpulan urutan perintah yang akan dijalankan oleh komputer. Pemrograman

prosedural merupakan salah satu contoh dari paradigma ini, dan seringkali dianggap sebagai sebuah paradigma yang sama.

- ⇒ Ide dasarnya adalah dari model komputer Von Neumann.
- ⇒ Eksekusi langkah-langkah komputasi diatur oleh sebuah struktur kontrol.
- ⇒ Berdasarkan urutan-urutan atau sekuensial.
- ⇒ Program adalah suatu rangkaian prosedur untuk memanipulasi data. Prosedur merupakan kumpulan instruksi yang dikerjakan secara berurutan.
- ⇒ Contoh bahasa pemrograman: Fortran, Algol, Pascal, Basic, C

1.6.2 Paradigma Fungsional

Pemrograman Fungsional adalah sebuah paradigma yang menjadikan fungsi matematika sebagai penentu dalam eksekusi komputasi. Fungsi tersebut merupakan dasar utama dari program yang akan dijalankan. Paradigma ini lebih banyak digunakan di kalangan akademis daripada produk komersial, terutama yang murni fungsional.

- ⇒ Ide dasar dari matematika dan teori fungsi.
- ⇒ Beberapa contoh bahasa fungsional adalah APL, Erlang, Haskell, Lisp, ML, Oz dan Scheme.

1.6.3 Paradigma Logika

Umumnya digunakan pada domain yang berhubungan dengan ekstraksi pengetahuan yang berbasis kepada fakta dan relasi. Dalam paradigma ini, logika digunakan secara murni untuk representasi bahasa deklaratif yang kebenarannya ditentukan oleh programmer, sedangkan pembukti-teorema atau model pembangkit digunakan sebagai pemecah masalah.

- ⇒ Berasal dari pembuktian otomatis didalam intelegensia buatan.
- ⇒ Berdasar kepada aksioma, aturan dan query.
- ⇒ Eksekusi program menjadi proses pencarian secara sistematis dalam sekumpulan fakta, dengan menggunakan sekumpulan aturan.
- ⇒ Beberapa contoh bahasa pemrograman: ALF, Fril, Gödel, Mercury, Oz, Ciao, Visual Prolog, XSB, and λ Prolog

1.6.4 Paradigma Berorientasi Obyek

Pemrograman berorientasi obyek muncul untuk mengatasi masalah kompleksitas dari sebuah perangkat lunak sehingga kualitas dari perangkat lunak tersebut dapat dikelola dengan lebih mudah. Caranya adalah dengan memperkuat *modularity* dan *reusability* didalam perangkat lunak tersebut. Pemrograman berorientasi obyek menggunakan obyek dan interaksi antar obyek dalam penyusunan sebuah perangkat lunak. Paradigma ini semakin banyak digunakan karena lebih mudah dalam menggambarkan kondisi yang ada pada dunia nyata.




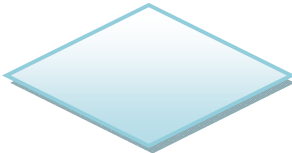





- ⇒ Ide dari interaksi antar obyek yang ada pada dunia nyata.
- ⇒ Antar obyek saling berinteraksi dengan saling mengirimkan pesan (*message*).
- ⇒ Terdapat beberapa karakteristik utama, yaitu: Abstraksi, Enkapsulasi, Pewarisan dan Polimorfisme.

1.7 Flowchart

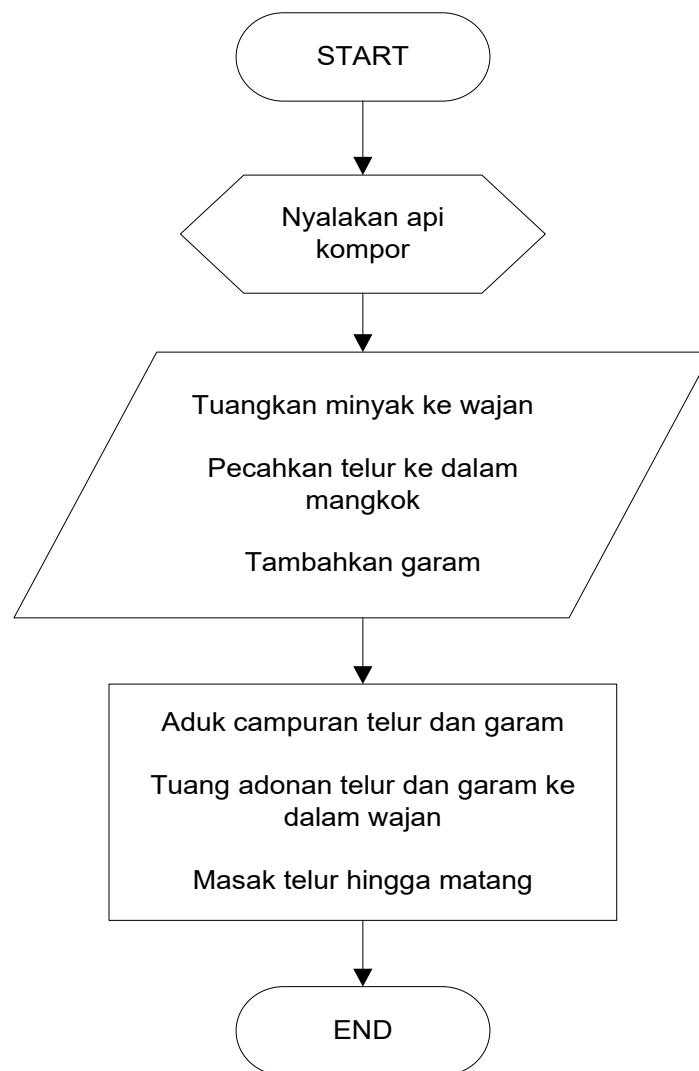
Dalam membuat algoritma, diperlukan suatu mekanisme atau alat bantu untuk menuangkan hasil pemikiran mengenai langkah-langkah penyelesaian masalah yang sistematis dan terurut. Pada dasarnya untuk bisa menyusun solusi diperlukan kemampuan *problem-solving* yang baik. Oleh karena itu, sebagai sarana untuk melatih kemampuan tersebut terdapat sebuah *tool* (alat) yang dapat digunakan, yakni *flowchart*.

Secara formal, *flowchart* didefinisikan sebagai skema penggambaran dari algoritma atau proses. Tabel berikut menampilkan simbol-simbol yang digunakan dalam menyusun *flowchart*.

Tabel 1.1 Simbol-simbol dalam *flowchart*

	<i>Terminator</i> Sebagai simbol 'START' atau 'END' untuk memulai atau mengakhiri flowchart.
	<i>Input/Output</i> Digunakan untuk menuliskan proses menerima data atau mengeluarkan data
	<i>Proses</i> Digunakan untuk menuliskan proses yang diperlukan, misalnya operasi aritmatika
	<i>Conditional / Decision</i> Digunakan untuk menyatakan proses yang membutuhkan keputusan
	<i>Preparation</i> Digunakan untuk memberikan nilai awal
	<i>Arrow</i> Sebagai penunjuk arah dan alur proses
	<i>Connector (On-page)</i> Digunakan untuk menyatukan beberapa arrow
	<i>Connector (Off-page)</i> Digunakan untuk menghubungkan flowchart yang harus digambarkan pada halaman yang berbeda. Biasanya pada simbol ini diberi nomor sebagai penanda, misalnya angka 1.
	<i>Display</i> Digunakan untuk menampilkan data ke <i>monitor</i>

Berikut ini adalah flowchart untuk menggambarkan kegiatan membuat telur dadar:



Gambar 1.2 Flowchart membuat telur dadar

Dengan menggunakan flowchart, tahapan-tahapan penting dalam algoritma dapat ditunjukkan dengan diagram di atas. Aliran proses ditunjukkan dengan arah panah atau disebut dengan 'flowlines'.

Keuntungan menggunakan *flowchart* adalah penggunaan diagram untuk menggambarkan tahapan proses, sehingga lebih mudah dilihat dan dipahami. Namun demikian, flowchart juga memiliki kelemahan, yakni jika digunakan untuk menggambarkan proses atau algoritma untuk skala kasus yang besar, maka akan dibutuhkan banyak kertas.

1.8 Pseudocode

Skema lain yang dapat digunakan untuk menyusun algoritma adalah *pseudocode*. *Pseudocode* adalah bentuk informal untuk mendeskripsikan algoritma yang mengikuti struktur bahasa pemrograman tertentu. Tujuan dari penggunaan *pseudocode* adalah supaya :

- ↳ lebih mudah dibaca oleh manusia
- ↳ lebih mudah untuk dipahami
- ↳ lebih mudah dalam menuangkan ide/hasil pemikiran

Pseudocode sering digunakan dalam buku-buku tentang ilmu komputer ataupun publikasi ilmiah untuk menjelaskan urutan proses atau metode tertentu. Seorang programmer yang ingin menerapkan algoritma tertentu, terutama yang kompleks atau algoritma baru, biasanya akan memulainya dengan membuat deskripsi dalam bentuk *pseudocode*. Setelah *pseudocode* tersebut jadi, maka langkah selanjutnya hanya tinggal menterjemahkannya ke bahasa pemrograman tertentu. *Pseudocode* ini biasanya disusun dalam bentuk yang terstruktur dengan pendekatan sekuensial (berurutan) dari atas ke bawah.

Algoritma yang menjelaskan tentang proses membuat telur dadar, sebenarnya sudah menerapkan penggunaan *pseudocode*. Sesungguhnya tidak ada suatu standar untuk menyusun algoritma menggunakan *pseudocode*.

Oleh karena *pseudocode* lebih cocok digunakan untuk menyusun algoritma dengan kasus yang besar dan kompleks, maka sangat dianjurkan kepada programmer pemula untuk mulai menggunakan *pseudocode* dalam menyelesaikan masalah. Berikut adalah contoh *pseudocode* yang dibandingkan dengan bahasa pemrograman C++.

Tabel 1.2 Notasi *pseudocode* dan bahasa C++

<i>Pseudocode</i>	<i>C++</i>
<pre> if sales > 1000 then bonus ← sales * 25% salary ← 2000000 + bonus endif output(salary) </pre>	<pre> int sales; sales=1001; if (sales > 1000) { bonus = sales * 0.25; salary = 2000 + bonus; } cout << "Salary : "<<salary; </pre>
<pre> If totalbelanja > 500000 then diskon ← 0.2 * totalbelanja bayar ← totalbelanja – diskon endif output(bayar) </pre>	<pre> int totalbelanja; totalbelanja =500001; if (totalbelanja > 500000) { diskon = 0.25 * totalbelanja; bayar = totalbelanja - diskon; } cout << "Bayar : "<<bayar; </pre>