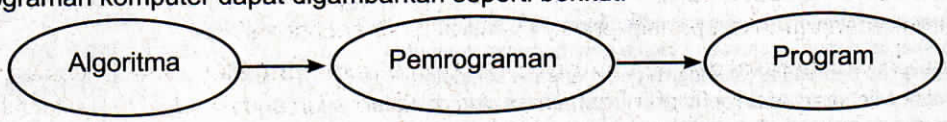
Program komputer merupakan sekumpulan instruksi berupa pernyataan yang tertulis dengan menggunakan bahasa pemrograman yang melibatkan pemilihan struktur data. Program komputer dihasilkan dari proses pemrograman oleh seorang programmer. Alur pemrograman meliputi definisi masalah, merumuskan solusi masalah, membuat algoritma, melakukan pengkodean, pengujian, dan dokumentasi.

Algoritma dibuat untuk membantu menyusun langkah-langkah penyelesaian masalah sesuai dengan logika. Algoritma menjadi acuan dalam pembuatan kode program nantinya. Pembuatan program menggunakan aplikasi bahasa pemrograman. Setiap bahasa pemrograman memiliki aturan sintaks yang berbeda-beda. Tahukah Anda hubungan antara logika dengan algoritma? Tahukah Anda yang dimaksud bahasa pemrograman? Pernahkah Anda menggunakan aplikasi bahasa pemrograman? Untuk mengetahui jawabannya, pelajari modul ini dengan saksama!

## A. Alur Pemrograman Komputer

Komputer adalah sekumpulan alat elektronik yang saling bekerja sama, dapat menerima data (input), mengolah data (proses), dan memberikan informasi (output), serta terkoordinasi di bawah kontrol program yang tersimpan di dalam memori. Program komputer merupakan sekumpulan instruksi berupa pernyataan yang tertulis dengan menggunakan bahasa pemrograman yang melibatkan pemilihan struktur data. Dalam arti luas, pemrograman meliputi seluruh kegiatan yang mencakup pembuatan, perencanaan, dan perwujudan program. Sedangkan dalam arti sempit, pemrograman merupakan pengkodean atau sering disebut dengan istilah *coding*, dan juga pengujian berdasarkan rancangan tertentu.

Alur pemrograman komputer dapat digambarkan seperti berikut.




Pada dasarnya, komputer memerlukan instruksi berupa langkah-langkah perintah sehingga sebuah prosedur dapat dijalankan. Prosedur yang berisi langkah-langkah penyelesaian masalah disebut algoritma. Agar instruksi-instruksi dalam algoritma dapat dimengerti (diproses) oleh komputer, maka harus diubah dalam bentuk program menggunakan bahasa pemrograman melalui proses pemrograman. Oleh karena itu, perancang harus menuliskan setiap segi dari permasalahan yang bersangkutan. Beberapa pakar komputer menyatakan bahwa program komputer terdiri atas algoritma dan bahasa pemrograman. Jadi sebelum memasuki tahap pemrograman komputer dengan bahasa pemrograman, sebaiknya mempelajari dahulu tentang logika dan algoritma pemrograman.

Seorang programmer tidak melakukan pengkodean program begitu saja, melainkan dengan mengikuti perencanaan terstruktur yang sistematis. Berikut langkah-langkah sistematis dasar dalam menyelesaikan masalah pemrograman.

**1. Definisi permasalahan**

Sebuah program biasanya dibuat berdasarkan sebuah permasalahan. Sebelum sebuah program dapat terdesain dengan baik untuk menyelesaikan sebuah permasalahan, masalah-masalah yang terjadi harus dapat diketahui dan terdefinisi dengan baik untuk mendapatkan detail persyaratan input dan output. Contoh pendefinisian masalah, yaitu "buatlah sebuah program yang akan menampilkan berapa kali sebuah nama tampil pada sebuah daftar".

**2. Analisis dan perumusan pemecahan masalah**

Setelah masalah terdefinisi, langkah selanjutnya adalah memecahkan masalah tersebut menjadi beberapa bagian kecil dan ringkas.

a. Contoh masalah : menampilkan jumlah kemunculan sebuah nama pada daftar.
b. Input program : daftar nama, nama yang akan dicari.
c. Output program : jumlah kemunculan nama yang dicari.

**3. Desain algoritma dan representasi**

Setelah mengetahui dengan jelas permasalahan yang ingin diselesaikan, langkah selanjutnya adalah membuat rumusan algoritma untuk penyelesaian masalah. Dalam pemrograman komputer, penyelesaian masalah didefinisikan dalam langkah demi langkah. Algoritma dapat direpresentasikan dalam bahasa manusia, *flowchart*, atau *pseudocode*.

**4. Pengkodean, pengujian, dan pembuatan dokumentasi**

Proses pengkodean (*coding*) program dapat dilakukan setelah pembuatan algoritma. *Coding* harus menggunakan algoritma yang telah dibuat sebagai pedoman agar program dapat dibuat sesuai rencana. Kode program dapat ditulis sesuai bahasa pemrograman yang dipilih. Apabila pengkodean sudah selesai, langkah selanjutnya adalah pengujian program tersebut apakah sudah berfungsi sesuai dengan tujuan program tersebut dibuat. Jika terdapat kesalahan-kesalahan atas logika program, disebut juga bug, maka program tersebut perlu untuk dikaji ulang rumusan atau algoritma yang telah dibuat, kemudian melakukan perbaikan implementasi kode program yang mungkin salah. Proses ini disebut *debugging*.Untuk memudahkan dalam memeriksa kesalahan suatu program ataupun untuk memahami jalannya suatu program, maka perlu dibuat sebuah dokumentasi dari program yang dibuat. Dokumentasi tersebut dapat berupa informasi mulai dari tujuan dan fungsi program, algoritma, serta cara penggunaannya.

## B. Logika dan Algoritma

Pembuatan algoritma tidak bisa lepas dari logika, yaitu kemampuan seorang manusia untuk berpikir dengan akal tentang suatu permasalahan yang menghasilkan sebuah kebenaran, dibuktikan, dan dapat diterima akal. Logika mutlak diperlukan dalam penyelesaian suatu masalah. Istilah logika identik dengan penalaran. Penalaran adalah salah satu bentuk pemikiran yang mungkin benar dan mungkin juga tidak benar. Secara sederhana, logika dapat didefinisikan sebagai ilmu yang memberikan prinsip-prinsip yang harus diikuti agar dapat berpikir valid menurut aturan yang berlaku. Pelajaran logika menimbulkan kesadaran untuk menggunakan prinsip-prinsip untuk berpikir secara sistematis.

Asal kata algoritma berasal dari nama **Abu Ja'far Mohammed Ibnu Musa Al-Khowarizmi**, ilmuwan Persia yang menulis kitab Al Jabr W'al-muqabala (*Rules of Restoration and Reduction*) sekitar tahun 825 M. Kata Al-Khowarizmi dibaca orang Barat menjadi *algorism,* lalu berubah menjadi *algorithm,* dan diserap dalam bahasa Indonesia menjadi algoritma.

Menurut **Rinaldi Munir**, algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis. Sedangkan menurut Kamus *Besar Bahasa Indonesia*, algoritma adalah urutan logis pengambilan keputusan untuk pemecahan masalah. Dari pengertian di atas dapat disimpulkan bahwa algoritma merupakan urutan langkah-langkah (instruksi-instruksi/aksi-aksi) terbatas yang disusun secara sistematis dan menggunakan bahasa yang logis untuk menyelesaikan suatu masalah.

Dalam hal ini, yang dimaksud menyelesaikan masalah adalah suatu permasalahan yang dipecahkan menggunakan komputer melalui sekumpulan instruksi yang dimengerti oleh komputer. Sekumpulan instruksi yang dimaksud adalah proses pembuatan program yang menggunakan alat bantu berupa bahasa pemrograman.

Contoh penggunaan logika dan algoritma untuk menghitung luas persegi panjang sebagai berikut.

1. Menentukan nilai panjang persegi panjang.
2. Menentukan nilai lebar persegi panjang.
3. Menghitung luas persegi panjang dengan cara mengalikan nilai panjang dengan nilai lebar.
4. Maka luas persegi panjang ditemukan.
5. Selesai.

Saat menggunakan logika, sebaiknya jangan berpikir terlalu rumit tentang sebuah masalah, karena belum tentu masalah itu serumit yang dipikirkan. Pikirkan hal yang paling sederhana untuk menyelesaikan masalah tersebut, sehingga tidak terjebak dalam pikiran yang rumit.

**1. Ciri-ciri dan kriteria algoritma**

Algoritma merupakan jantung ilmu komputer atau informatika, tetapi algoritma dapat ditemukan dalam kehidupan sehari-hari. Contohnya cara membuat masakan yang dinyatakan dalam suatu resep dapat disebut sebagai algoritma, karena dalam resep masakan terdapat urutan langkah-langkahnya. Bila langkah-langkah tersebut tidak logis, maka tidak dapat menghasilkan makanan yang diinginkan. Namun, tidak semua urutan langkah penyelesaian masalah yang logis dapat disebut sebagai algoritma.

Menurut **Donal E.Knuth,** algoritma mempunyai ciri-ciri sebagai berikut.

a. *Finiteness* (keterbatasan), yaitu algoritma harus berakhir setelah melakukan sejumlah langkah proses.
b. *Definiteness* (kepastian), yaitu setiap langkah algoritma harus didefinisikan dengan tepat dan tidak menimbulkan makna ganda.
c. Input (masukan), yaitu sebuah algoritma memiliki nol atau lebih masukan (input) yang diberikan algoritma sebelum dijalankan.
d. Output (keluaran), yaitu setiap algoritma memberikan satu atau beberapa hasil keluaran.
e. *Effectiveness* (efektivitas), yaitu langkah algoritma dikerjakan dalam waktu yang wajar.

Sedangkan sifat algoritma sebagai berikut.

a. Tidak menggunakan simbol atau sintaks dari suatu bahasa pemrograman tertentu.
b. Tidak tergantung pada suatu bahasa pemrograman tertentu.
c. Notasi-notasinya dapat digunakan untuk seluruh bahasa mana pun.
d. Algoritma dapat digunakan untuk merepresentasikan suatu urutan kejadian secara logis dan dapat diterapkan di semua kejadian sehari-hari.

Pembuatan algoritma harus memenuhi kriteria berikut.

a. Setiap langkah harus jelas dan pasti (*definite*).
b. Diperbolehkan tanpa ada input, tetapi minimal harus ada satu output.
c. Jumlah langkah harus berhingga atau dengan kata lain harus ada *stopping criteria*.
d. Urutan langkah-langkah untuk memecahkan masalah adalah urutan langkah logis (urutan langkah logis berarti algoritma harus mengikuti suatu urutan tertentu dan tidak boleh melompat-lompat).

**2. Aturan penulisan algoritma**

Algoritma seseorang bisa berbeda dari algoritma orang lain dalam menyelesaikan suatu pekerjaan. Alur pemikiran dapat dituangkan secara tertulis, artinya bisa berupa kalimat, gambar, atau tabel tertentu. Di bidang komputer, algoritma sangat diperlukan dalam menyelesaikan berbagai masalah pemrograman, terutama dalam komputasi numeris. Tanpa algoritma yang dirancang dengan baik, maka proses pemrograman akan menjadi salah, rusak, atau lambat, dan tidak efisien. Penulisan algoritma dapat menggunakan notasi berikut.

a. Menggunakan bahasa natural (bahasa manusia: contoh bahasa Indonesia atau bahasa Inggris), adalah masih sering membingungkan (ambigu)/sulit dipahami.
b. Menggunakan *flowchart*, baik digunakan karena alur algoritma dapat dilihat secara visual, tetapi kesulitannya dalam pembuatannya jika algoritma panjang.
c. Menggunakan *pseudocode*, sudah dekat dengan bahasa pemrograman, tetapi sulit dimengerti oleh orang yang belum memahami pemrograman.

Pembuatan program yang baik harus memenuhi tiga kriteria, yaitu *correct* (benar), *clear* (jelas), dan *efficient* (efisien). Agar program memenuhi tiga kriteria di atas, maka harus mengetahui tahapan-tahapan penulisan algoritma berikut.

| Tahapan Algoritma | Keterangan |
|---|---|
| Merencanakan algoritma | Mendefinisikan masalah dan merencanakan garis besar pemecahannya. Contohnya permasalahan membuat aplikasi menghitung luas persegi panjang. Garis besar pemecahannya bahwa untuk menghitung luas persegi panjang dihitung dengan rumus luas = panjang x lebar di mana input yang diperlukan adalah panjang dan lebar persegi panjang. |
| Menyatakan suatu algoritma | Menerjemahkan permasalahan menjadi langkah-langkah pemecahan masalah yang urut dan logis. |
| Alat yang digunakan untuk menulis algoritma | Contoh untuk menyelesaikan permasalahan di atas, langkah-langkahnya:<br>1. Masukkan panjang dan lebar<br>2. Hitung luas = panjang x lebar<br>3. Cetak luas |

| Tahapan Algoritma | Keterangan |
|---|---|
| Bahasa natural | Penulisan alogaritma menggunakan bahasa natural, menerjemahkan langkah-langkah tersebut dengan menggunakan bahasa sehari-hari. |
| *Flowchart* | Dengan menggunakan simbol-simbol *flowchart*. |
| *Pseudocode* | Suatu teknik penulisan algoritma dengan menggunakan sebanyak mungkin komponen-komponen dari salah satu bahasa tingkat tinggi (suatu bahasa pemrograman yang masih memerlukan unit kompilator untuk mengeksekusi program agar dapat berjalan). Penulisan algoritma dengan *pseudocode* hampir menyerupai sebuah program, tetapi tanpa menyertakan atribut-atribut program (seperti tipe data dan lain-lain), hanya menuliskan proses intinya saja. Aturan penulisan *pseudocode* sebagai berikut.<br>**Judul algoritma:** bagian yang terdiri atas nama algoritma dan penjelasan (spesifikasi) tentang algoritma tersebut. Nama sebaiknya singkat dan menggambarkan apa yang dilakukan oleh algoritma tersebut.<br>**Deklarasi:** bagian untuk mendefinisikan semua nama yang digunakan di dalam program. Nama tersebut berupa nama konstanta, variabel, tipe, prosedur, dan fungsi.<br>**Deskripsi:** bagian yang berisi uraian langkah-langkah penyelesaian masalah. Contohnya permasalahan menghitung luas persegi panjang berikut.<br>`Algoritma LuasPersegiPanjang` (Judul algoritma digunakan untuk menghitung luas persegi panjang dengan input panjang, lebar)<br>`DEKLARASI`<br>`Panjang,lebar:int`<br>`DESKRIPSI`<br>`Read(panjang,lebar)`<br>`Luas = Panjang * lebar`<br>`Write(Luas)`<br>`End.` |
| Menerjemahkan algoritma ke dalam bahasa pemrograman (pengkodean/*coding*) | Kode/program adalah pernyataan-pernyataan yang dituliskan dalam bahasa pemrograman.<br>Perhatikan contoh menerjemahkan algoritma ke dalam bahasa pemrograman berikut!<br>**Menggunakan bahasa pemrograman Pascal**<br>`Program latihan1;`<br>`var panjang,lebar:integer`<br>`begin`<br>`write('Masukkan panjang=');readln(panjang);`<br>`write('Masukkan lebar=');readln(lebar)`<br>`Luas:=panjang*lebar;`<br>`writeln('Luas Persegi panjang=',luas);`<br>`end.` |

## C. Algoritma Menggunakan Bahasa Natural

Dalam kehidupan sehari-hari sering ditemui algoritma dengan menggunakan bahasa natural (bahasa sehari-hari). Bahasa yang digunakan tentunya bahasa yang mudah dipahami oleh pengguna algoritma. Misalnya petunjuk penggunaan laptop, jika laptop tersebut dijual di Indonesia tentu akan disediakan petunjuk dalam bahasa Indonesia. Jika dijual di beberapa negara dengan bahasa yang berbeda-beda biasanya petunjuk akan disajikan dalam banyak bahasa (multilingual) karena algoritma yang baik adalah algoritma yang dapat dipahami dan dijalankan oleh penggunanya.

Penyajian algoritma dalam bahasa natural menggunakan kalimat deskriptif, yaitu menjelaskan secara detail suatu algoritma dengan bahasa atau kata-kata yang mudah dipahami. Penyajian algoritma ini cocok untuk algoritma yang singkat, namun sulit untuk algoritma yang besar. Selain itu, algoritma ini akan sulit dikonversi ke bahasa pemrograman.

Contoh dari penyajian algoritma menggunakan bahasa natural sebagai berikut.

1. Menentukan nilai jari-jari (r) lingkaran.
2. Menentukan nilai phi, yaitu 3,14.
3. Menghitung luas lingkaran dengan cara mengalikan nilai jari-jari (r) dengan nilai jari-jari (r) lalu dikalikan dengan nilai phi.
4. Maka luas lingkaran ditemukan.
5. Selesai.

## D. *Pseudocode*

*Pseudocode* merupakan salah satu metode penulisan algoritma selain menggunakan bahasa natural. Berasal dari kata *pseudo* dan *code*, *pseudocode* memiliki arti kode semu atau menyerupai kode program yang sebenarnya. *Pseudocode* menggunakan simbol-simbol yang mirip atau menyerupai kode program yang ditulis dengan menggunakan bahasa pemrograman tertentu. Selain itu, biasanya *pseudocode* menggunakan bahasa yang mudah dipahami secara universal dan juga lebih ringkas daripada algoritma dengan bahasa natural.

**Contoh:**

```
Algoritma Keliling Persegi Panjang
{Algoritma yang akan digunakan untuk menghitung keliling persegi panjang}
Deklarasi
    panjang     : integer
    lebar       : integer
    keliling    : integer
Algoritma
    read (panjang)
    read (lebar)
    keliling ← 2*(panjang+lebar)
    write (keliling)
    end
```

Perhatikan contoh perbedaan antara algoritma menggunakan bahasa natural dan *pseudocode* dengan contoh algoritma mencari luas persegi panjang berikut!

| Bahasa Natural | *Pseudocode* |
|---|---|
| Masukkan panjang | Input panjang |
| Masukkan lebar | Input lebar |
| Nilai luas adalah panjang x lebar | Luas ←panjang x lebar |
| Tampilkan luas | Print luas |

## E. *Flowchart*

*Flowchart* merupakan penulisan algoritma menggunakan simbol-simbol yang memang sudah menjadi standar pada dunia komputer. *Flowchart* digunakan untuk menggambarkan alur algoritma menggunakan diagram dengan tujuan memperjelas langkah-langkah penyelesaian algoritma. Dalam pembuatan *flowchart,* pengguna bisa menggunakan aplikasi Microsoft Word atau bisa pula menggunakan aplikasi Microsoft Visio.

*Flowchart* dapat dibagi menjadi dua, yaitu sebagai berikut.

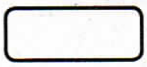
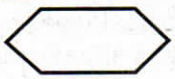
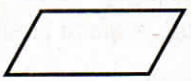
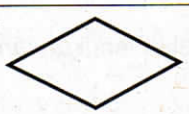
1. **Sistem *flowchart***

   Sistem *flowchart* adalah *flowchart* yang melukiskan urutan operasi pemrosesan data, baik yang manual maupun dengan komputer. Diberikan pula aliran data, yaitu file yang digunakan selama proses dan pengontrolannya.

2. **Program *flowchart***

   Program *flowchart* adalah suatu bagan dengan simbol-simbol tertentu yang menggambarkan urutan proses secara mendetail dan hubungan antara suatu proses (instruksi) dengan proses lainnya dalam suatu program.

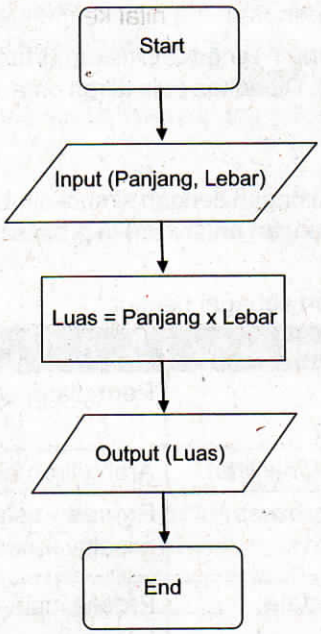
   Tabel aturan simbol pada *flowchart* sebagai berikut.

| Simbol | Nama | Fungsi |
|---|---|---|
|  | Terminator | Permulaan atau akhir program/algoritma |
|  | Garis aliran (*flow line*) | Arah aliran algoritma program |
|  | *Predefined process* | Proses inisialisasi atau pemberian harga awal, misalnya mendefinisikan konstanta di awal |
|  | *Input/output* data | Proses input atau output data |
| | *Process* | Proses perhitungan atau pengolahan data |
|  | *Decision* (percabangan) | Kondisi bersyarat, yang memungkinkan algoritma menjalankan perintah untuk kondisi yang berbeda-beda |
| | *ON page connector* | Penghubung bagian *flowchart* yang berada pada satu halaman yang berbeda digunakan jika algoritma cukup besar sehingga harus dipecah penyajiannya |
| | *OFF page connector* | Penghubung bagian *flowchart* yang berada pada satu halaman yang berbeda, digunakan jika algoritma cukup besar sehingga harus dipecah penyajiannya |
| | *Display* | Mencetak keluaran dalam layar monitor |
| | *Punche Card* | Menyatakan input berasal dari kartu atau output ditulis ke kartu |

Pada pembuatan *flowchart* terdapat beberapa pedoman, antara lain:

a. *Flowchart* sebaiknya digambar dari atas ke bawah dan dimulai dari bagian kiri suatu halaman.
b. Kegiatan di dalam *flowchart* harus ditunjukkan dengan jelas.
c. Harus ditunjukkan dari mana kegiatan dimulai dan di mana akan berakhir.
d. Masing-masing kegiatan di dalam *flowchart* sebaiknya menggunakan suatu kata yang mewakili suatu pekerjaan.
e. Masing-masing kegiatan di dalam *flowchart* harus di dalam urutan yang tepat.
f. Kegiatan yang terpotong dan akan disambungkan ke tempat lain harus ditunjukkan dengan jelas menggunakan simbol penghubung yang benar.
g. Gunakan simbol-simbol aliran yang standar.

Berikut contoh *flowchart* penyajian algoritma menghitung luas persegi panjang.




## F. Struktur Algoritma

Seperti yang kita ketahui, penyusun atau struktur dasar sebuah algoritma adalah langkah-langkah. Berikut penjelasan ringkas mengenai tiga struktur dasar algoritma.

**1. Runtunan (*sequential*)**

Setiap instruksi atau perintah dalam runtunan ditulis dalam satu baris atau beberapa instruksi dalam baris yang sama, tetapi antara setiap instruksi dipisahkan dengan tanda *semicolon*/titik koma (;). Beberapa karakteristik runtunan, antara lain:

a. Setiap instruksi dikerjakan satu per satu.
b. Setiap instruksi hanya dikerjakan satu kali, tidak ada instruksi yang diulang.
c. Urutan instruksi yang ditulis dalam notasi algoritma sama dengan urutan instruksi yang akan diproses.
d. Akhir dari instruksi terakhir adalah akhir dari algoritma.

**Contoh 1:**

Perhatikan contoh operasi aritmatika berikut!

(2+5)*7=49, akan tetapi apabila urutan aksinya diubah, maka hasil keluaran akan berbeda misalnya menjadi 2+(5*7)=37.

**Contoh 2:**

Perhatikan contoh algoritma bahasa natural runtunan aksi penukaran dua bilangan bulat berikut!

a. Deklarasikan A, B, dan C sebagai bilangan bulat.
b. Masukkan nilai A dan nilai B.
c. Masukkan nilai A ke dalam C.
d. Masukkan nilai B ke dalam A.
e. Masukkan nilai C ke dalam B.

**Contoh 3:**

Perhatikan contoh alogaritma *pseudocode* pertukaran dua bilangan berikut!

a. Nilai pertama (A) dan nilai kedua (B) sudah terdefinisi.
b. Menampilkan hasil pertukaran dua bilangan (A dan B).

Deklarasi:

A, B : integer {variabel nilai pertama dan nilai kedua}

C : integer {variabel bantu}

Algoritma:

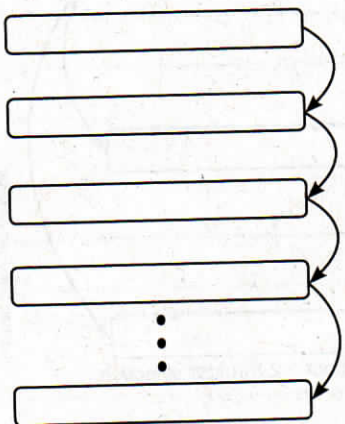
C ← A

A ← B

B ← C

Output ("Nilai Pertama = ", B)

Output ("Nilai Kedua = ", A)



Sumber: *Dokumen penerbit*

Gambar 1.1 Struktur *sequential*

**2. Pemilihan (*selection*)**

Sering kali suatu instruksi hanya bisa dikerjakan jika suatu kondisi tertentu telah terpenuhi, sehingga komputer tidak lagi mengerjakan instruksi secara berurutan seperti pada runtunan, melainkan berdasarkan kondisi yang terpenuhi. Pemilihan adalah instruksi yang dikerjakan dengan kondisi tertentu. Kondisi adalah persyaratan yang dapat bernilai benar atau salah. Beberapa instruksi hanya dilaksanakan apabila kondisi bernilai benar, sebaliknya apabila kondisi bernilai salah maka instruksi tidak dilaksanakan.

Berikut tiga jenis kasus pada struktur pemilihan.

a. Analisis satu kasus (IF-THEN)
b. Analisis dua kasus (IF-THEN-ELSE)
c. Analisis tiga kasus atau lebih (IF-THEN-ELSE bertingkat dan struktur CASE)

**Contoh 1:**

Perhatikan contoh algoritma bahasa natural penentuan bilangan genap atau ganjil berikut!

a. Masukkan sebuah bilangan bulat.
b. Bagi bilangan dengan angka 2, simpan nilai sisa pembagian dalam variabel sisa.
c. Jika nilai sisa sama dengan 0, maka kerjakan langkah d.
d. Tampilkan "GENAP" ke layar.
e. Jika nilai sisa tidak sama dengan 0, maka kerjakan langkah f.
f. Tampilkan "GANJIL" ke layar.
g. Selesai.

**Contoh 2:**

Perhatikan contoh alogaritma *pseudocode* menentukan bilangan positif berikut!

a. User memasukkan sebuah nilai.
b. Menampilkan hasil proses apakah nilai yang dimasukkan user adalah bilangan positif atau bukan.

Deklarasi:

X : real

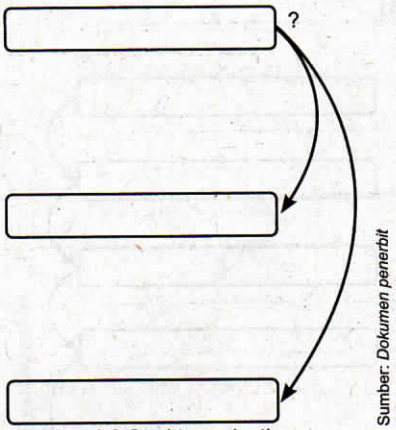
Algoritma:

Input (X)

IF X > 0 THEN

Output (X, "adalah bilangan positif")

Output (X, "bukan bilangan positif")




Sumber: *Dokumen penerbit*

Gambar 1.2 Struktur *selection*

**3. Perulangan (*repetition* atau *loop*)**

Salah satu kelebihan komputer adalah kemampuannya untuk mengerjakan pekerjaan atau perintah yang sama berulang kali tanpa mengenal lelah dan bosan. Perulangan adalah kegiatan mengerjakan sebuah atau sejumlah aksi yang sama sebanyak jumlah yang ditentukan atau sesuai kondisi yang diinginkan. Berikut tiga jenis konstruksi pada struktur perulangan.

a. FOR NEXT
b. WHILE WEND
c. DO LOOP

**Contoh 1:**

Perhatikan contoh algoritma bahasa natural menampilkan angka tertentu sebanyak n kali ke layar berikut!

a. Deklarasikan variabel angka untuk menyimpan angka yang akan ditampilkan.
b. Deklarasikan variabel n untuk menyimpan banyaknya perulangan.
c. Deklarasikan variabel counter yang digunakan sebagai counter perulangan yang sudah dilakukan.
d. Masukkan sebuah angka dan simpan dalam variabel angka.
e. Masukkan banyaknya perulangan yang diinginkan dan simpan dalam variabel n.
f. Set nilai counter dengan 0.
g. Tampilkan angka ke layar.
h. Lakukan penambahan counter dengan 1.
i. Jika nilai counter<n, kerjakan langkah 6.
j. Jika nilai counter=n, maka selesai.

**Contoh 2:**

Perhatikan contoh alogaritma *pseudocode* mencetak nilai 1 sampai n berikut!

a. User memasukkan nilai terakhir (N).
b. Mencetak nilai dari 1 sampai N.

Deklarasi:

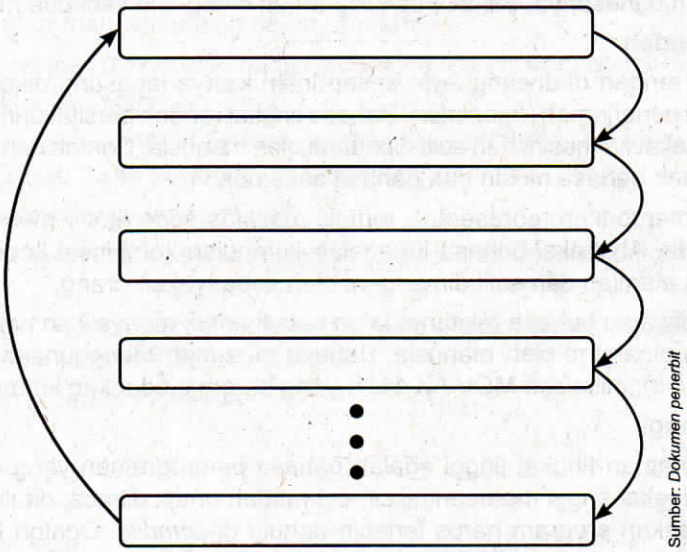
X : real

Algoritma:

Input (N)

I ← 1

WHILE i ≤ N DO



Sumber: *Dokumen penerbit*

Gambar 1.3 Struktur *loop*

## G. Bahasa Pemrograman

Pembuatan algoritma digunakan untuk membantu mengonversikan suatu permasalahan ke dalam bahasa pemrograman. Algoritma merupakan hasil pemikiran konseptual sehingga agar dilaksanakan oleh komputer, maka algoritma harus diterjemahkan ke dalam notasi bahasa pemrograman. Ada beberapa hal yang harus diperhatikan dalam penerjemahan ke dalam bahasa pemrograman, yaitu:

1. **Pendeklarasian variabel**

   Variabel digunakan untuk menyimpan data masukan, memproses data, dan mendapatkan hasil komputasi.

2. **Pemilihan tipe data**

   Jika dalam pembuatan program membutuhkan pendeklarasian variabel, maka kita harus memilih tipe data, karena setiap variabel pasti membutuhkan tipe data ketika dideklarasikan.

3. **Pemakaian atau pemilihan instruksi**

   Urutan langkah-langkah dalam algoritma dapat diterjemahkan menjadi salah satu atau beberapa instruksi algoritma, yaitu runtunan, pemilihan, atau perulangan.

4. **Aturan sintaksis**

   Setiap bahasa pemrograman mempunyai aturan penulisan sintaks tersendiri. Jadi, penulisan sintaks program harus disesuaikan dengan bahasa pemrograman yang digunakan.

5. **Tampilan hasil**

   Pada saat membuat algoritma kita tidak memikirkan hasil tampilan yang akan disajikan. Tampilan ini akan diperhatikan ketika mengonversikannya menjadi program.

6. **Cara pengoperasian compiler atau interpreter**

   Bahasa pemrograman yang digunakan termasuk dalam kelompok compiler atau interpreter. Masing-masing bahasa pemrograman memiliki cara pengoperasian compiler/interpreter yang berbeda-beda.

Bahasa pemrograman merupakan suatu sistem komunikasi yang berupa instruksi-instruksi standar untuk memerintah komputer agar menjalankan fungsi tertentu. Kumpulan instruksi ini harus dimengerti oleh komputer, berstruktur tertentu (sintaksis), dan bermakna. Penulisan bahasa pemrograman sebagai alat komunikasi untuk memberikan perintah kepada komputer tidak berlaku kebebasan berekspresi seperti layaknya dalam menulis dalam bahasa alamiah. Penulisan bahasa pemrograman harus mengikuti aturan sintaksis masing-masing bahasa pemrograman yang digunakan. Bahasa pemrograman yang umum digunakan adalah HTML/CSS, JavaScript, PHP, C++, C#, C, Python, Java, Ruby, dan Visual Basic.

Berdasarkan kerumitan bahasanya, bahasa pemrograman dibagi menjadi dua macam, yaitu:

**1. Bahasa tingkat rendah**

Bahasa tingkat rendah dirancang agar setiap instruksinya langsung dikerjakan oleh komputer, tanpa harus melalui penerjemah (*translator*). Bahasa tingkat rendah bersifat primitif, sangat sederhana, orientasinya lebih dekat ke mesin, dan sulit dipahami oleh manusia. Contoh dari bahasa pemrograman tingkat rendah adalah bahasa mesin dan bahasa assembly.

Bahasa mesin merupakan representasi tertulis *machine code* (kode mesin), yaitu kode operasi suatu mesin tertentu. Abstraksi bahasa ini adalah kumpulan kombinasi kode biner 0 dan 1, yang berarti sangat tidak alamiah dan sulit dimengerti oleh kebanyakan orang.

Bahasa assembly atau bahasa rakitan adalah notasi untuk menyajikan bahasa mesin yang lebih mudah dibaca dan dipahami oleh manusia. Bahasa ini sudah menggunakan simbol alfabet yang bermakna (mnemonic), misalnya MOV AX 1111, yang berarti pindahkan ke register AX nilai 1111.

**2. Bahasa tingkat tinggi**

Bahasa pemrograman tingkat tinggi adalah bahasa pemrograman yang dekat dengan bahasa manusia. Bahasa tingkat tinggi mempunyai ciri-ciri mudah untuk dibaca, ditulis, maupun diperbarui, sebelum bisa dijalankan program harus terlebih dahulu di-*compile*. Contoh bahasa pemrograman tingkat tinggi adalah Ada, BASIC, COBOL, C, C++, FORTRAN, Pascal, dan sebagainya.

Bahasa pemrograman tingkat tinggi harus di-*compile* menggunakan compiler. Compiler adalah suatu program yang menerjemahkan bahasa program (*source code*) ke dalam bahasa objek (*object code*). Compiler memerlukan waktu untuk membuat suatu program yang dapat dieksekusi oleh komputer. Akan tetapi program yang diproduksi dengan compiler akan berjalan lebih cepat dibandingkan dengan program yang diproduksi dengan interpreter. Interpreter menganalisis dan mengeksekusi setiap baris dari program tanpa melihat program secara keseluruhan. Tanpa melalui kompilasi, interpreter biasanya digunakan pada pembuatan program berskala besar.

Untuk lebih memahami perbedaan antara bahasa tingkat rendah dengan bahasa tingkat tinggi, perhatikan tabel berikut!

| Bahasa Tingkat Rendah | Bahasa Tingkat Tinggi |
|---|---|
| Lebih mendekati bahasa mesin. | Lebih mendekati bahasa manusia. |
| Berhubungan dengan register, alamat memori, dan stack-stack panggilan. | Berhubungan dengan variabel, larik, dan ekspresi aritmatika, atau aljabar boolan. |
| Cenderung untuk membuat kode yang lebih efisien. | Cenderung untuk membuat pemrograman yang kompleks menjadi lebih sederhana. |
| Dirancang agar setiap instruksinya langsung dikerjakan oleh komputer, tanpa harus melalui penerjemah (*translator*). | Program tidak dapat langsung dilaksanakan oleh komputer, harus diterjemahkan terlebih dahulu oleh sebuah *translator* bahasa yang biasa disebut compiler, ke dalam bahasa mesin. |

Berikut ini penjelasan beberapa bahasa pemrograman beserta contoh penggunaannya.

**1. Bahasa Pascal**

Bahasa pemrograman Pascal pertama kali dirancang oleh **Prof. Nicklaus Wirth** dari *Technical University* di Zurich, Switzerland tahun 1971. Nama Pascal berasal dari **Blaise Pascal**, nama ahli matematika dan filsafat dari Prancis (abad ke-17). Bahasa Pascal merupakan pengembangan dari bahasa Algol 60 dan Algol W (turunan Algol 60), memiliki beberapa versi, seperti Turbo Pascal, Ms Pascal (Microsoft), Apple Pascal, UCSD (*University of California at San Diego Pascal*), dan lain-lain.

Turbo Pascal yang dibuat oleh Borland Inc. adalah versi yang paling banyak digunakan karena menggunakan compiler untuk menerjemahkannya dan juga mengikuti standar bahasa Pascal yang dibuat oleh **Nicklaus Wirth** dan **K. Jensen**. Pascal merupakan bahasa pemrograman tingkat tinggi (*high level language*) dan terstruktur (*structured programming language*).

a. Struktur program Pascal

Struktur program Pascal terdiri dari bagian-bagian seperti berikut.

1) Judul program (*program heading*) bersifat *optional* (boleh digunakan/tidak), tetapi sebaiknya digunakan karena mencantumkan nama program.
2) Blok program (*program block*) atau badan program (*program body*), terdiri dari dua bagian berikut.
   a) Bagian deklarasi (*declaration part*)

      Untuk menyiapkan elemen-elemen program, seperti nama konstanta, variabel, label, tipe, prosedur, fungsi, serta penggunaan unit.
   b) Bagian pernyataan (*statement part*)

      Untuk menunjukkan suatu tindakan yang akan dikerjakan oleh program. Diawali Begin dan diakhiri end.

Pada pemrograman Pascal setiap akhir pernyataan diakhiri tanda titik koma (;), kecuali untuk nama label dan akhir program diberi tanda titik (.).

Perhatikan contoh program Pascal berikut!

```
program lat1;
uses crt;
label Akhir;
const Phi = 3.14;
type Bil_Nyata = Real;
var Jari_jari : Bil_Nyata;
procedure Hitung_Luas(Radius:Bil_Nyata);
begin
writeln ('Luas=',0.5*Phi*Radius*Radius);
end;
begin
clrscr;
jari_jari:= 10;
hitung_Luas ( jari_jari );
goto Akhir;
writeln ('Lho,kok saya dilewati!');
akhir:
writeln ('Selesai' );
end.
```

b. Komentar

Komentar digunakan untuk dokumentasi atau memberikan keterangan, dan baris program ini tidak akan dieksekusi atau diabaikan. Penulisan komentar boleh menggunakan tanda berikut.

```
{ ini komentar } atau (* ini komentar *)
```

Perhatikan contoh komentar berikut!

```
{Ini awal Program}
begin
writeln ('Bahasa');
writeln ('Pascal');
end. (*Akhir Program*)
```

c. Perintah input dan output

Perintah input dan output dalam program Pascal terdiri dari perintah-perintah berikut.

1) Perintah read dan readln
   a) Digunakan untuk meminta masukan dari keyboard untuk diolah komputer.
   b) Tipe data yang dicetak bisa berupa integer, real, character, string, ataupun boolean.
   c) Perbedaan read dan readln adalah setelah meminta masukan. Jika readln akan diakhiri dengan pindah baris, maka pada read tidak.
2) Perintah write dan writeln
   a) Digunakan untuk mencetak hasil proses. Tipe data yang dicetak bisa berupa integer, real, character, string, ataupun boolean.
   b) Perbedaan antara write dan writeln adalah setelah mencetak. Jika writeln akan diakhiri dengan pindah baris, maka pada write tidak.

**2. Bahasa Delphi**

Bahasa Borland Delphi atau sering disebut Delphi saja adalah sarana pemrograman aplikasi visual. Bahasa pemrograman yang digunakan adalah bahasa pemrograman Pascal atau selanjutnya disebut bahasa pemrograman Delphi. Delphi merupakan generasi penerus dari Turbo Pascal.

Kelebihan bahasa Delphi, antara lain sebagai berikut.

a. Sifatnya *freeware*.
b. Dikembangkan dengan bahasa Pascal, sehingga bagi pengguna yang terbiasa dengan dasar pemrograman Turbo Pascal akan lebih familiar.
c. Komponen yang disediakan sudah cukup lengkap tanpa harus menambahkan komponen dari sumber lain.
d. Dokumentasi cukup lengkap.

Kelemahan bahasa Delphi antara lain sebagai berikut.

a. Pengguna yang tidak memiliki dasar pemrograman bahasa Pascal akan mengalami kesulitan untuk pertama kalinya.
b. Setiap komponen yang dimasukkan dalam form tampilan akan diikutsertakan kode deklarasi dan inisialisasinya dalam *list code*. Sehingga apabila terjadi perubahan komponen, penamaan, maupun kesalahan penulisan kode, program tidak mau membetulkan secara otomatis.

**3. Bahasa C**

Tahun 1978, **Brian W. Kerninghan** dan **Dennis M. Ritchie** dari *AT & T Laboratories* mengembangkan bahasa B menjadi bahasa C. Bahasa B yang diciptakan oleh **Ken Thompson** sebenarnya merupakan pengembangan dari bahasa BCPL (*Basic Combined Programming Language*) yang diciptakan oleh **Martin Richard**. Sejak tahun 1980, bahasa C banyak digunakan *programmer* di Eropa yang sebelumnya menggunakan bahasa B dan BCPL. Dalam perkembangannya, bahasa C menjadi bahasa paling populer di antara bahasa lainnya, seperti Pascal, BASIC dan FORTRAN. Tahun 1989, dunia pemrograman C mengalami peristiwa penting dengan dikeluarkannya standar bahasa C oleh *American National Standards Institute* (ANSI).

Bahasa C yang diciptakan **Kerninghan** dan **Ritchie** kemudian dikenal dengan nama ANSI C. Mulai awal tahun 1980, **Bjarne Stroustrup** dari *AT & T Bell Laboratories* mulai mengembangkan bahasa C. Pada tahun 1985, lahirlah secara resmi bahasa baru hasil pengembangan C yang dikenal dengan nama C++.

Sebenarnya bahasa C++ mengalami dua tahap evolusi, yaitu C++ yang pertama, dirilis oleh *AT&T Laboratories*, dinamakan cfront. C++ versi kuno ini hanya berupa compiler yang menerjemahkan C++ menjadi bahasa C. Pada evolusi selanjutnya, Borland International Inc. mengembangkan compiler C++ menjadi sebuah compiler yang mampu mengubah C++ langsung menjadi bahasa mesin (*assembly*). Sejak evolusi ini, mulai tahun 1990 C++ menjadi bahasa berorientasi objek yang digunakan oleh sebagian besar *programmer* profesional.

a. Struktur bahasa C

Kerangka program bahasa C sebagai berikut.

```
/* Nama File : .....*/
/*identitas perancang/penulis */
/* deskripsi ringkas dari program */
<tipe> main([int argc, char** argv[,char ** envp]]) {
/*Keterangan program*/
/*Kamus*/
/*Algoritma/deretan instruksi yang dieksekusi */
Return (<retval>) ;
}
```

Keterangan:

1) Ada satu atau lebih jenis fungsi, namun yang harus ada di dalam sebuah program C adalah fungsi main ( ) yang menandai awal dan akhir eksekusi program.
2) Return value main akan diberikan ke lingkungan yang menjalankan program (biasanya berupa angka integer, yang menyatakan tingkat kesalahan yang terjadi saat terminasi program). Kebanyakan program tidak mengembalikan nilai sehingga deklarasi yang digunakan adalah void main () { program; }.
3) Parameter argc merupakan parameter bertipe int, menyatakan jumlah argumen yang diberikan pada program saat dipanggil (nama program dianggap sebagai argumen, sehingga jumlah argumen minimum adalah 1).
4) Parameter argv merupakan pointer ke string yang akan menyimpan parameter-parameter apa saja yang digunakan dalam eksekusi program.
5) Parameter envp adalah pointer ke array string lingkungan (environtment), akhir array ditandai dengan NULL.

Perhatikan contoh program bahasa C berikut!

**Contoh:**

```
// my first program in C
#include <stdio.h>
int main ()
{
  Printf ("Hello World!";
  return 0;
}
```

Hasil atau output:

```
Hello World!
```

b. Komentar

Komentar boleh menggunakan tanda: // ini komentar atau /* ini komentar */

```
/* Ini awal Program */
int main(){
  printf ("ayo belajar C");
  return 1;
}/* Akhir Program */
```

Hasil atau output:

```
ayo belajar C
```

c. Perintah input dan output

Perintah input dan output dalam program bahasa C terdiri atas perintah-perintah berikut.

1) Perintah scanf
   a) Digunakan untuk membaca masukan dari keyboard untuk diolah komputer.
   b) Tipe data yang dibaca dapat berupa integer, real, character, string, ataupun boolean.

2) Perintah printf
   a) Digunakan untuk mencetak hasil proses.
   b) Tipe data yang dicetak dapat berupa integer, real, character, string, ataupun boolean.

**Tabel Format Tipe Data**

| Kode Format | Tipe Data |
|---|---|
| %d | Long<br>Int<br>Short |
| %f | Float<br>Double<br>Long double |
| %c | Char |
| %s | Char<br>Nama_variabel<br>[] |

Perhatikan contoh program dengan menggunakan bahasa C berikut!

**Contoh:**

```
#include <stdio.h>
/* Program menghitung luas persegi panjang */
int main(){
int panjang, lebar, luas;
printf("Panjang=");
scanf("%d",&panjang);
printf("Lebar=");
scanf("%d",&lebar);
luas=panjang*lebar;
printf("Luas Persegi Panjang= %d",luas);
}
```

Program di atas digunakan untuk menghitung luas persegi panjang. Penjelasannya sebagai berikut.

int panjang, lebar, luas : variabel yang dideklarasikan adalah panjang, lebar, dan luas dengan tipe data integer.

printf ("Panjang") : mencetak tulisan "Panjang=" ke layar untuk meminta masukan nilai panjang dari pengguna.

scanf ("%d", &panjang) : menyimpan nilai masukan dari pengguna ke dalam variabel panjang.

Printf ("Lebar=") : mencetak tulisan "Lebar=" ke layar untuk meminta masukan nilai lebar dari pengguna.

scanf ("%d", &lebar) : menyimpan nilai masukan dari pengguna ke dalam variabel lebar.

luas=panjang*lebar : menghitung luas persegi panjang dengan mengalikan nilai panjang dengan nilai lebar dan menyimpan nilainya dalam variabel luas.

printf ("Luas Persegi Panjang=%d", luas) : mencetak ke layar hasil perhitungan luas persegi panjang.

d. Library bahasa C

Daftar library yang dimiliki bahasa C, yaitu sebagai berikut.

1) stdio.h untuk input data terformat [scanf(), gets()] dan output [printf()].
2) conio.h untuk input data tidak terformat [getche( ), getchar( ), getch( ), dan gets( )].
3) string.h untuk manipulasi data string.
4) stdlib.h, library standar dari C.
5) math.h untuk fungsi matematika lanjut.
6) ctype.h, library utility pendukung.
7) bool.h untuk menggunakan true/false.

## 4. Bahasa C++

Bahasa C++ merupakan bahasa pemrograman komputer yang dikembangkan oleh **Bjarne Stroustrup** di Bell Labs pada awal tahun 1980-an berdasarkan C ciptaan ANSI (*American National Standard Institute*). Prototype C++ muncul pertama kali sebagai bahasa C yang dipercanggih dengan fasilitas class, atau biasa disebut C dengan class (*C with class*). Selama tahun 1983-1984, C dengan class disempurnakan dengan menambah fasilitas baru yang kemudian melahirkan bahasa C++. Simbol "++" berarti operator penaikan yang menunjukkan bahwa bahasa ini merupakan versi yang lebih canggih dari bahasa C.

Salah satu perbedaan mendasar antara bahasa C dengan C++ adalah dukungan terhadap konsep pemrograman berorientasi objek. Meskipun bahasa C dan C++ menggunakan sintaks yang sama, tetapi mereka memiliki perbedaaan, yaitu C merupakan bahasa pemrograman prosedural, di mana penyelesaian suatu masalah dilakukan dengan membagi-bagi masalah tersebut ke dalam submasalah yang lebih kecil. Sedangkan C++ sebagai bahasa pemrograman yang memiliki sifat pemrograman berorientasi objek, untuk menyelesaikan masalah dilakukan dengan menjelaskan class-class yang merupakan anak class yang dibuat sebelumnya sebagai abstraksi dari objek-objek fisik. Class tersebut berisi keadaan objek, anggota-anggotanya, dan kemampuan dari objeknya. Setelah beberapa class dibuat, kemudian masalah dipecahkan dengan class.

Berikut ini contoh perintah input dan output dalam bahasa C++.

a. Perintah cout

Penggunaan cout dihubungkan dengan operator *overloaded* << (sepasang tanda "*less than*"). Perhatikan contoh penggunaan perintah cout berikut!

**Contoh:**

```
cout << "Output sentence"; // mencetak tulisan "Output sentence" di layar
cout << 120;              // mencetak angka 120 di layar
cout << x;                // mencetak variabel x di layar
```

Operator << dikenal sebagai *insertion* operator, yang berfungsi untuk menginput data yang mengikutinya. Jika berupa string, maka harus diapit dengan tanda kutip ganda ("), sehingga membedakannya dengan variabel.

**Contoh:**

```
cout << "Hello"; // mencetak tulisan "Hello" di layar
cout << Hello;   // mencetak variabel Hello di layar
```

Operator *insertion* (<<) dapat digunakan lebih dari satu kali dalam baris yang sama.

**Contoh:**

```
cout << "Hello, " << "I am " << "a C++ sentence";
```

Contoh di atas akan menampilkan tulisan "Hello, I am a C++ sentence" pada layar monitor. Manfaat dari pengulangan penggunaan operator *insertion* (<<) adalah untuk menampilkan kombinasi dari satu variabel dan konstanta atau lebih.

**Contoh:**

```
cout << "Hello, I am " << age << " years old and my zipcode is " << zipcode;
```

b. Perintah cin

Perintah cin menangani standar input pada C++ dengan menambahkan *overloaded operator extraction* (>>) dan harus diikuti dengan variabel yang akan menyimpan data.

**Contoh:**

```
int age;
cin >> age;
```

Contoh di atas mendeklarasikan variabel age dengan tipe int dan menunggu input dari cin (keyboard) untuk disimpan di variabel age. Perintah cin akan memproses input dari keyboard sekali saja dan tombol ENTER harus ditekan.

**Contoh:**

```
// i/o example
#include <iostream.h>
int main ()
{
  int i;
  cout << "Please enter an integer value: ";
  cin >> i;
  cout << "The value you entered is " << i;
  cout << " and its double is " << i*2 << ".\n";
  return 0;
}
```

**Output** :

```
Please enter an integer value: 702
The value you entered is 702 and its double is 1404.
```

Perintah cin juga dapat digunakan untuk lebih dari satu input yaitu seperti berikut.

```
cin >> a >> b;
```

ekuivalen dengan:

```
cin >> a;
cin >> b;
```

Dalam hal ini data yang di-input harus dua, satu untuk variabel a dan lainnya untuk variabel b yang penulisannya diketik melalui keyboard.

Penulisan komentar pada bahasa C++ sama dengan bahasa C, yaitu tulisan dianggap komentar setelah tanda // atau terletak di antara tanda /* dan */.

Untuk memahami perbedaan penulisan kode program menggunakan bahasa C dan bahasa C++, perhatikan contoh program dengan bahasa C dan bahasa C++ berikut!

**Contoh program dengan bahasa C:**

```
#include <stdio.h>
int main()
{
  int jam, besar, bayar;
  printf ("Berapa jam kamu bekerja?");
  scanf ("%d",&jam);
  printf ("Berapa besar gaji tiap jamnya?");
  scanf ("%d",&besar);
  bayar=jam*besar;
  printf ("Kamu mendapatkan gaji Rp%d", bayar);
return 0;
}
```

**Contoh program dengan C++:**

```
#include <iostream.h>
int main()
{
  int jam, besar, bayar;
  cout << "Berapa jam kamu bekerja?";
```

```
    cin >> jam;
    cout << "Berapa besar gaji tiap jamya?";
    cin >> besar;
    bayar=jam*besar;
    cout << "Kamu mendapatkan gaji Rp" <<bayar<<end;
    return 0;
}
```

**5. Bahasa Visual Basic**

Bahasa BASIC pada dasarnya adalah bahasa yang mudah dimengerti sehingga pemrograman di dalam bahasa BASIC dapat dengan mudah dilakukan meskipun oleh orang yang baru belajar membuat program. Visual Basic adalah bahasa pemrograman yang sangat mudah dipelajari, dengan teknik pemrograman visual yang memungkinkan penggunanya untuk berkreasi lebih baik dalam menghasilkan suatu program aplikasi.

Bahasa Visual Basic dikembangkan oleh Microsoft sejak tahun 1991 yang merupakan pengembangan dari bahasa pemrograman BASIC (*Beginner's All-purpose Symbolic Instruction Code*) yang dikembangkan pada era 1950-an. Visual Basic merupakan salah satu development tool, yaitu alat bantu untuk membuat berbagai program komputer, khususnya yang menggunakan sistem operasi Windows. Visual Basic merupakan salah satu bahasa pemrograman komputer yang berorientasi objek (*Object Oriented Programming*/OOP).

Kelebihan:

a. Bahasa yang sederhana.
b. Karena Visual Basic sangat populer, sangat banyak sumber-sumber yang dapat digunakan untuk belajar dan mengembangkan kemampuan *programming*.
c. Bisa memperoleh *tools* baik yang gratis maupun berbayar di internet, sehingga sangat membantu menghemat waktu dalam membuat program.

Kelemahan:

a. Visual Basic adalah bahasa pemrograman yang *powerful*, tetapi sebenarnya tidak terlalu bagus untuk membuat game-game yang benar-benar memuaskan.
b. Lebih lambat dibandingkan bahasa pemrograman lainnya.

## H. Microsoft Visual Basic

Pada modul ini kita akan membahas tentang bahasa pemrograman Visual Basic. Bahasa pemrograman ini sederhana sehingga dapat dibuat dengan cepat dan mudah. Kita bisa membuat program dengan aplikasi GUI (*Graphical User Interface*) atau program yang memungkinkan pemakai komputer berkomunikasi dengan komputer tersebut dengan menggunakan modus grafis atau gambar. Sebelum melakukan pemrograman dengan bahasa pemrograman Visual Basic, terlebih dahulu kita perlu melakukan instalasi perangkat lunak bahasa pemrogramannya yaitu Microsoft Visual Basic.

Microsoft Visual Basic 6.0 menyediakan fasilitas yang memungkinkan kita untuk menyusun sebuah program dengan memasang objek-objek grafis dalam sebuah form. Visual Basic 6.0 berawal dari bahasa pemrograman BASIC (*Beginners All-purpose Symbolic Instruction Code*). Karena bahasa BASIC cukup mudah dipelajari dan populer, maka hampir setiap programmer menguasai bahasa ini.

BASIC pertama kali dikembangkan pada tahun 1963 oleh **John George Kemeny** dan **Thomas Eugene Kurtz** yang berasal dari Darthmouth College, untuk menginzinkan akses terhadap komputer bagi para mahasiswa jurusan selain jurusan ilmu eksakta. Pada waktu itu hanya matematikawan dan ilmuwan yang dapat menggunakan komputer karena mereka dapat membuat perangkat lunak sendiri.