BAB I

Menjelaskan Struktur Algoritma

A. Pengantar Algoritma Dan Program

1. Apakah Itu Algoritma

Ditinjau dari asal-usul katanya, kata Algoritma sendiri mempunyai sejarah yang aneh. Orang hanya menemukan kata algorism yang berarti proses menghitung dengan angka arab. Anda dikatakan algorist jika Anda menghitung menggunakan angka arab. Para ahli bahasa berusaha menemukan asal kata ini namun hasilnya kurang memuaskan. Akhirnya para ahli sejarah matematika menemukan asal kata tersebut yang 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). Dari judul buku itu kita juga memperoleh akar kata "Aljabar" (Algebra). Perubahan kata dari 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, maka lambat laun kata algorithm berangsur-angsur dipakai sebagai metode perhitungan (komputasi) secara umum, sehingga kehilangan makna kata aslinya. Dalam bahasa Indonesia, kata algorithm diserap menjadi algoritma.

(Sumber: www.wikipedia.org)

Gambar 1. (penemu algoritma)

2. Algoritma Merupakan Jantung Ilmu Informatika

Algoritma adalah jantung ilmu komputer atau informatika. Banyak cabang ilmu komputer yang mengarah ke 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. Cara-cara membuat kue atau masakan yang dinyatakan dalam suatu resep juga dapat disebut sebagai algoritma. Pada setiap resep selalu ada urutan langkah-langkah membuat masakan. Bila langkah-langkahnya tidak logis, tidak dapat dihasilkan masakan yang diinginkan. Ibu-ibu yang mencoba suatu resep masakan akan membaca satu per satu langkah-langkah pembuatannya lalu ia mengerjakan proses sesuai yang ia baca. Secara umum, pihak (benda) yang mengerjakan proses disebut pemroses (processor). Pemroses tersebut dapat berupa manusia, komputer, robot atau alat-alat elektronik lainnya. Pemroses melakukan suatu proses dengan melaksanakan atau “mengeksekusi” algoritma yang menjabarkan proses tersebut.

B. Definisi Algoritma

“Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis dan logis”. Kata logis merupakan kata kunci dalam algoritma. Langkah-langkah dalam algoritma harus logis dan harus dapat ditentukan bernilai salah atau benar.

Dalam beberapa konteks, algoritma adalah spesifikasi urutan langkah untuk melakukan pekerjaan tertentu. Pertimbangan dalam pemilihan algoritma adalah, pertama, algoritma haruslah benar. Artinya algoritma akan memberikan keluaran yang dikehendaki dari sejumlah masukan yang diberikan. Tidak peduli sebagus apapun algoritma, kalau memberikan keluaran yang salah, pastilah algoritma tersebut bukanlah algoritma yang baik Pertimbangan kedua yang harus diperhatikan adalah kita harus mengetahui seberapa baik hasil yang dicapai oleh algoritma tersebut. Hal ini penting terutama pada algoritma untuk menyelesaikan masalah yang memerlukan aproksimasi hasil (hasil yang hanya berupa pendekatan). Algoritma yang baik harus mampu memberikan hasil yang sedekat mungkin dengan nilai yang sebenarnya.

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

menyelesaikan suatu permasalahan, walaupun terjadi perbedaan dalam menyusun algoritma, tentunya kita mengharapkan keluaran yang sama. Jika terjadi demikian, carilah algoritma yang paling efisien dan cepat.

Perhatikan algoritma sederhana berikut :

Jika seseorang ingin mengirim surat kepada kenalannya di tempat lain, langkah yang harus dilakukan adalah:

1. Menyiapkan Peralatan Tulis
2. Menulis surat
3. Surat dimasukkan ke dalam amplop tertutup
4. Amplop ditempeli perangko secukupnya.
5. Pergi ke Kantor Pos terdekat untuk mengirimkannya

Algoritma menghitung luas persegi panjang:

1. Masukkan panjang (P)
2. Masukkan lebar (L)
3. Luas ← P * L
4. Tulis Luas

1. **Pembuatan algoritma mempunyai banyak keuntungan di antaranya:**
   a) Pembuatan atau penulisan algoritma tidak tergantung pada bahasa pemrograman manapun, artinya penulisan algoritma independen dari bahasa pemrograman dan komputer yang melaksanakannya.
   b) Notasi algoritma dapat diterjemahkan ke dalam berbagai bahasa pemrograman.
   c) Apapun bahasa pemrogramannya, output yang akan dikeluarkan sama karena algoritmanya sama.

2. **Beberapa hal yang perlu diperhatikan dalam membuat algoritma:**
   a) Teks algoritma berisi deskripsi langkah-langkah penyelesaian masalah. Deskripsi tersebut dapat ditulis dalam notasi apapun asalkan mudah dimengerti dan dipahami.
   b) Tidak ada notasi yang baku dalam penulisan teks algoritma seperti notasi bahasa pemrograman. Notasi yang digunakan dalam menulis algoritma disebut notasi algoritmik.
   c) Setiap orang dapat membuat aturan penulisan dan notasi algoritmik sendiri. Hal ini dikarenakan teks algoritma tidak sama dengan teks program. Namun, supaya notasi algoritmik mudah ditranslasikan ke dalam notasi bahasa

pemrograman tertentu, maka sebaiknya notasi algoritmik tersebut berkorespondensi dengan notasi bahasa pemrograman secara umum.

d) Notasi algoritmik bukan notasi bahasa pemrograman, karena itu pseudocode dalam notasi algoritmik tidak dapat dijalankan oleh komputer. Agar dapat dijalankan oleh komputer, pseudocode dalam notasi algoritmik harus ditranslasikan atau diterjemahkan ke dalam notasi bahasa pemrograman yang dipilih. Perlu diingat bahwa orang yang menulis program sangat terikat dalam aturan tata bahasanya dan spesifikasi mesin yang menjalannya. *pseudocode* adalah kode yang mirip dengan instruksi kode program sebenarnya.

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

f) Algoritma merupakan hasil pemikiran konseptual, supaya dapat dilaksanakan oleh komputer, algoritma harus ditranslasikan ke dalam notasi bahasa pemrograman.

C. Definisi Program / Pemrograman

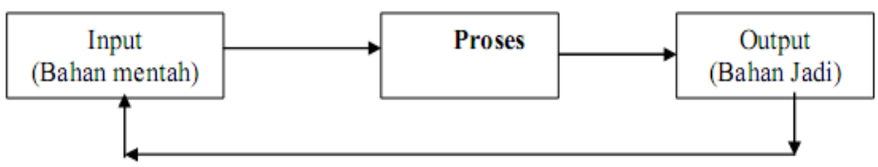
Pemrograman adalah kumpulan instruksi-instruksi tersendiri yang biasanya disebut source code yang dibuat oleh programmer (pembuat program)

Program adalah kumpulan instruksi atau perintah yang disusun sedemikian rupa sehingga mempunyai urutan nalar yang tepat untuk menyelesaikan suatu persoalan. (Menurut P. Insap Santosa)

Bahasa Pemrograman adalah alat untuk membuat program

Contoh: *C, C++, C#, Pascal, Basic, Perl, PHP, ASP, JHP, Java, dll.*

Secara garis besar, unsur-unsur pemrograman adalah Input →Proses → Output.



Gambar 2. Unsur Pemrograman

1. **Input**

   Bagian ini merupakan proses untuk memasukkan data ke komputer melalui device yang ada misalnya keyboard, mouse, scanner dll. Program melakukan proses membaca data yang akan diolah dari device tersebut.

2. **Output**

   Bagian ini merupakan proses untuk menampilkan data yang telah diolah, melaporkan hasil pegolahan data melalui device seperti monitor, printer dll. Program melakukan proses mencetak data ke device tersebut.

3. **Proses**

   Bagian ini merupakan proses mengolah data yang diinputkan dengan nenerapkan metode-metode, teknik-teknik, algoritma-algoritma yang ada. Proses ini menghasilkan data output yang akan dioutputkan kepada pengguna program.

Beda Algoritma dan Program ?

Program adalah kumpulan pernyataan komputer, sedangkan metode dan tahapan sistematis dalam program adalah algoritma. Program ditulis dengan menggunakan bahasa pemrograman. Jadi bisa disebut bahwa program adalah suatu implementasi dari bahasa pemrograman.

Beberapa pakar memberi formula bahwa:

Program = Algoritma + Bahasa (Struktur Data)

Penerjemah Bahasa Pemrograman

Untuk menterjemahkan bahasa pemrograman yang kita tulis maka diperlukan *Compiler* dan *interpreter*.

Compiler adalah suatu program yang menterjemahkan bahasa program (Source code) ke dalam bahasa obyek (object code) secara keseluruhan program.

Interpreter berbeda dengan *Compiler, interpreter* menganalisis dan mengeksekusi setiap baris dari program secara keseluruhan. Keuntungan dari interpreter adalah dalam eksekusi yang bisa dilakukan dengan segera. Tanpa melalui tahap kompilasi, untuk alas an ini interpreter digunakan pada saat pembuatan program berskala besar.

Perbedaan Compiler dan interpreter.

Compiler	*Interpreter*
Menterjemahkan secara keseluruhan	Menterjemahkan Instruksi per instruksi
Bila terjadi kesalahan kompilasi maka source program harus diperbaiki dan dikompilasi ulang	Bila terjadi kesalahan interprestasi dapat diperbaiki

Dihasilkan Object program	Tidak dihasilkan obyek program
Dihasilkan Executable program	Tidak dihasilkan Executable program
Proses pekerjaan program lebih cepat	Proses pekerjaan program lebih lambat
Source program tidak dipergunakan hanya bila untuk perbaikan saja	Source program terus dipergunakan
Keamanan dari program lebih terjamin	Keamanan dari program kurang terjamin

Paradigma Pemrograman :

1) Pemrograman Prosedural
   a) Berdasarkan urutan-urutan, sekuensial
   b) Program adalah suatu rangkaian prosedur untuk memanipulasi data. Prosedur merupakan kumpulan instruksi yang dikerjakan secara berurutan.
   c) Harus mengingat prosedur mana yang sudah dipanggil dan apa yang sudah diubah.
2) Pemrograman Fungsional
   a) Berdasarkan teori fungsi matematika
   b) Fungsi merupakan dasar utama program.
3) Pemrograman Terstruktur
   a) Secara berurutan dan terstrukrtur.
   b) Program dapat dibagai-bagi menjadi prosedur dan fungsi.
   c) Contoh: *PASCAL dan C*
4) Pemrograman Modular
   a) Pemrograman ini membentuk banyak modul.
   b) Modul merupakan kumpulan dari prosedur dan fungsi yang berdiri sendiri
   c) Sebuah program dapat merupakan kumpulan modul-modul.
   d) Contoh: MODULA-2 atau ADA
5) Pemrograman Berorientasi Obyek
   a) Pemrograman berdasarkan prinsip obyek, dimana obyek memiliki data/variabel/property dan method/event/prosedur yang dapat dimanipulasi
   b) Contoh: *C++, Object Pascal, dan Java*.
6) Pemrograman Berorientasi Fungsi
   a) Pemrograman ini berfokus pada suatu fungsi tertentu saja. Sangat tergantung pada tujuan pembuatan bahasa pemrograman ini.
   b) Contoh: *SQL (Structured Query Language), HTML, XML* dan lain-lain.
7) Pemrograman Deklaratif
   a) Pemrograman ini mendeskripsikan suatu masalah dengan pernyataan daripada memecahkan masalah dengan implementasi algoritma.
   b) Contoh: *PROLOG*

D. Belajar Memprogram dan Belajar Bahasa Pemrograman

Belajar memprogram tidak sama dengan belajar bahasa pemrograman. Belajar memprogram adalah belajar tentang metodologi pemecahan masalah, kemudian menuangkannya dalam suatu notasi tertentu yang mudah dibaca dan dipahami. Sedangkan belajar bahasa pemrograman berarti belajar memakai suatu bahasa aturan-aturan tata bahasanya, pernyataan-pernyataannya, tata cara pengoperasian compiler-nya, dan memanfaatkan pernyataan-pernyataan tersebut untuk membuat program yang ditulis hanya dalam bahasa itu saja. Sampai saat ini terdapat puluhan bahasa pemrogram, antara lain bahasa rakitan *(assembly), Fortran, Cobol, Ada, PL/I, Algol, Pascal, C, C++, Basic, Prolog, LISP, PRG,* bahasa-bahasa simulasi seperti *CSMP, Simscript, GPSS, Dinamo.*

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

1. **Bahasa pemrograman bertujuan khusus**. Yang termasuk kelompok ini adalah Cobol (untuk terapan bisnis dan administrasi). Fortran (terapan komputasi ilmiah), bahasa rakitan (terapan pemrograman mesin), Prolog (terapan kecerdasan buatan), bahasa-bahasa simulasi, dan sebagainya.
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 bahasa pemrograman dikelompokkan atas dua macam:

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. Sedangkan bahasa rakitan dimasukkan ke dalam kelompok ini karena alasan notasi yang dipakai dalam bahasa ini lebih dekat ke mesin, meskipun untuk melaksanakan instruksinya masih perlu penerjemahan ke dalam bahasa mesin.

2. **Bahasa tingkat tinggi,** yang membuat pemrograman lebih mudah 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 (yang disebut kompilator atau compiler) ke dalam bahasa mesin sebelum akhirnya dieksekusi oleh CPU. Contoh bahasa tingkat tinggi adalah Pascal, PL/I, Ada, Cobol, Basic, Fortran, C, PHP, C++, dan sebagainya.

Latihan 1:

1. Buatlah Algoritma Penerimaan Siswa Baru SMK N 1 Kawunganten ?
2. Buatlah Algoritma Mengitung Luas Segitiga ?
3. Buatlah Algoritma menjahit pakaian ?
4. Buatlah Algoritma membuat nasi goreng?
5. Jelaskan perbedaan Algoritma dan pemrograman?
6. Sebutkan Keuntungan Pembuatan algoritma ?
7. Berdasarkan pada notasi bahasa pemrograman lebih “dekat” ke mesin atau ke bahasa manusia, maka bahasa pemrograman dapat dikelompokkan menjadi, Sebutkan & Jelaskan ?
8. Berdasarkan pada notasi bahasa pemrograman lebih “dekat” ke mesin atau ke bahasa manusia, maka bahasa pemrograman dapat dikelompokkan menjadi ? Sebutkan & Jelaskan?
9. Sebutkan dan Jelaskan paradigma pemrograman ?

BAB II

Membuat Alur Logika Pemograman

A. Penyajian atau Penulisan Algoritma

Penyajian algoritma secara garis besar bisa dalam 2 bentuk penyajian yaitu tulisan dan gambar. Algoritma yang disajikan dengan tulisan yaitu dengan struktur bahasa tertentu (misalnya bahasa Indonesia atau bahasa Inggris) dan pseudocode. Pseudocode adalah kode yang mirip dengan kode pemrograman yang sebenarnya seperti Pascal, atau C, sehingga lebih tepat digunakan untuk menggambarkan algoritma yang akan dikomunikasikan kepada pemrogram. Sedangkan algoritma disajikan dengan gambar, yaitu dengan Flowchart

B. Flowchart (Diagram Alir)

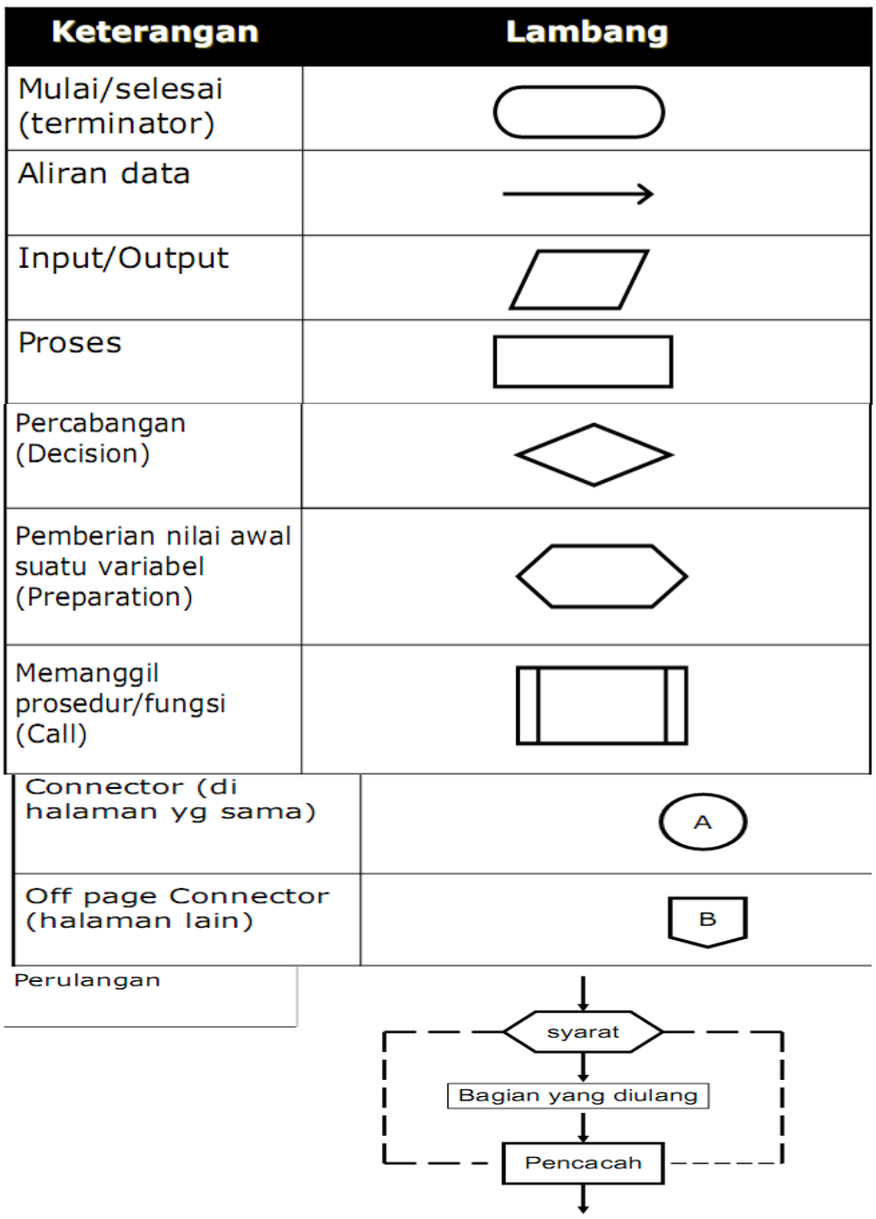
Flowchart atau bagan alir adalah skema/bagan (chart) yang menunjukkan aliran (flow) di dalam suatu program secara logika.

Flowchart merupakan alat yang banyak digunakan untuk menggambarkan algoritma dalam bentuk notasi-notasi tertentu. Flowchart merupakan gambar atau bagan yang memperlihatkan urutan dan hubungan antar proses beserta pernyataannya. Gambaran ini dinyatakan dengan simbol. Dengan demikian setiap simbol menggambarkan proses tertentu. Sedangkan antara proses digambarkan dengan garis penghubung. Dengan menggunakan flowchart akan memudahkan kita untuk melakukan pengecekan bagian-bagian yang terlupakan dalam analisis masalah. Di samping itu flowchart juga berguna sebagai fasilitas untuk berkomunikasi antara pemrogram yang bekerja dalam tim suatu proyek.

Walaupun tidak ada kaidah-kaidah yang baku dalam penyusunan flowchart, namun ada beberapa anjuran:

1) Hindari pengulangan proses yang tidak perlu dan logika yang berbelit sehingga jalannya proses menjadi singkat.
2) Jalannya proses digambarkan dari atas ke bawah dan diberikan tanda panah untuk memperjelas.
3) Sebuah flowchart diawali dari satu titik START dan diakhiri dengan END.

Berikut merupakan beberapa contoh simbol flowchart yang disepakati oleh dunia pemrograman:



Gambar 3. Simbol Flowchart

Pengertian Simbol Flowchart:

1) **Terminator** : Notasi ini digunakan untuk menunjukkan awal dan akhir suatu algoritma
2) **Aliran data :** Notasi ini disebut Arrow yang digunakan untuk menunjukkan arus data atau aliran data dari proses satu ke proses lainnya.
3) **Input / Output** : Notasi ini disebut Data yang digunakan untuk mewakili data input atau output atau menyatakan operasi pemasukan data dan pencetakan hasil.

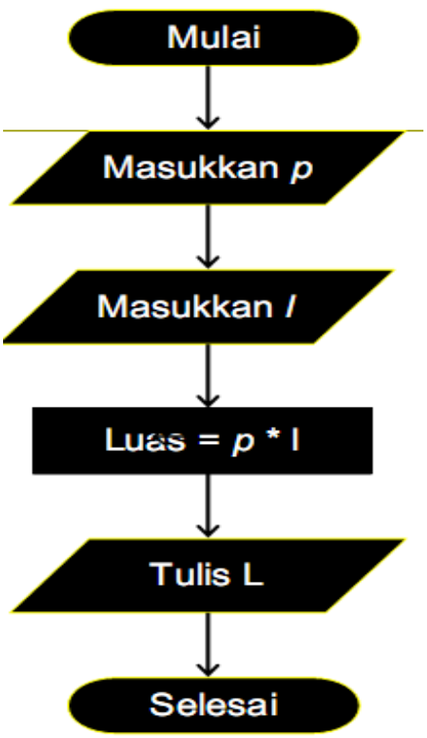
4) **Proses :** Notasi ini disebut Process yang digunakan untuk mewakili suatu proses.
5) **Percabangan :** Notasi ini disebut Decision yang digunakan untuk suatu pemilihan, penyeleksian kondisi di dalam suatu program
6) **Preparation :** Notasi ini digunakan untuk memberi nilai awal, nilai akhir, penambahan / pengurangan bagi suatu variable counter.
7) **Predefined Process / Call :** Notasi ini digunakan untuk menunjukkan suatu operasi yang rinciannya ditunjukkan ditempat lain (prosedur, sub-prosedur, fungsi)
8. **Connector :** Notasi ini digunakan untuk menunjukkan sambungan dari flowchart yang terputus di halaman yang sama atau halaman berikutnya.
9. O**ff page Connector :** Penghubung bagian-bagian flowchart yang berada pada halaman berbeda

Contoh Flowchart :

Problem: Menghitung luas persegi panjang

Algoritma:

1. Masukkan panjang (p)
2. Masukkan lebar (l)
3. Hitung luas (L), yaitu panjang kali lebar
4. Cetak luas (L)



Algoritma Pemrograman :

```
<?php
$panjang =6;
$lebar =5;
$luas= $panjang*$lebar;
echo "Jadi Luas Persegi Panjang adalah . $luas";
?>
```

C. Struktur Dasar Algoritma

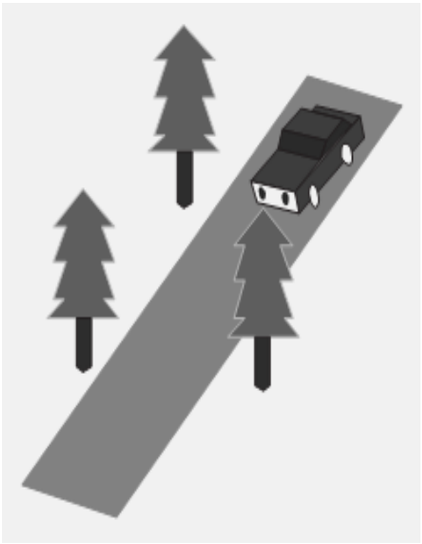
Algoritma berisi langkah-langkah penyelesaian suatu masalah. Langkah-langkah tersebut dapat berupa runtunan aksi (sequence), pemilihan aksi (selection),

pengulangan aksi (iteration) atau kombinasi dari ketiganya. Jadi struktur dasar pembangunan algoritma ada tiga, yaitu:

1) Struktur Runtunan / Beruntun : Digunakan untuk program yang pernyataannya sequential atau urutan.
2) Struktur Pemilihan / Percabangan : Digunakan untuk program yang menggunakan pemilihan atau penyeleksian kondisi.
3) Struktur Perulangan : Digunakan untuk program yang pernyataannya akan dieksekusi berulang-ulang.

1. Struktur Algoritma Runtunan / Berurutan :

Ada tiga struktur dasar yang digunakan dalam membuat algoritma yaitu struktur berurutan (sequencing), struktur pemilihan/keputusan/percabangan (branching) dan struktur pengulangan (looping). Sebuah algoritma biasanya akan menggabungkan ketiga buah struktur ini untuk menyelesaikan masalah. Pada bagian ini kita akan bahas lebih dulu struktur algoritma berurutan. Struktur berurutan dapat kita samakan dengan mobil yang sedang berjalan pada jalur lurus yang tidak terdapat persimpangan seperti tampak pada Gambar disamping Mobil tersebut akan melewati kilometer demi kilometer jalan sampai tujuan tercapai. *Struktur berurutan terdiri satu atau lebih instruksi.*

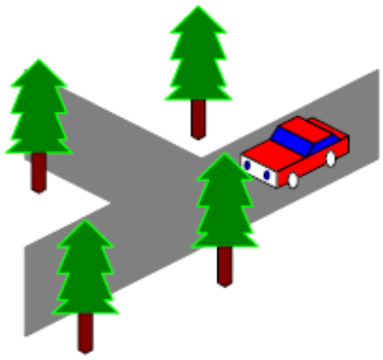


Gambar 4. Algoritma Runtunan

Tiap instruksi dikerjakan secara berurutan sesuai dengan urutan penulisannya, yaitu sebuah instruksi dieksekusi setelah instruksi sebelumnya selesai dieksekusi. Urutan instruksi menentukan keadaan akhir dari algoritma. Bila urutannya diubah, maka hasil akhirnya mungkin juga berubah. Menurut Goldshlager dan Lister (1988) struktur berurutan mengikuti ketentuan-ketentuan sebagai berikut:

a. tiap instruksi dikerjakan satu persatu
b. tiap instruksi dilaksanakan tepat sekali, tidak ada yang diulang
c. urutan instruksi yang dilaksanakan pemroses sama dengan urutan aksi sebagaimana yang tertulis di dalam algoritmanya
d. akhir dari instruksi terakhir merupakan akhir algoritma.

2. Struktur Algoritma Percabangan

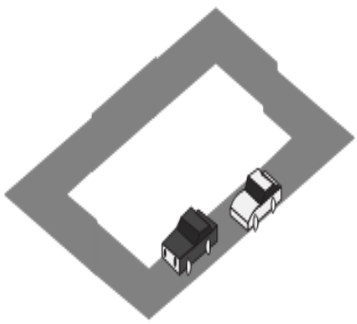


Gambar 5. Algoritma Percabangan

Sebuah program tidak selamanya akan berjalan dengan mengikuti struktur berurutan, kadang-kadang kita perlu merubah urutan pelaksanaan program dan menghendaki agar pelaksanaan program meloncat ke baris tertentu. Peristiwa ini kadang disebut sebagai percabangan/pemilihan atau keputusan.

Pada struktur percabangan, program akan berpindah urutan pelaksanaan jika suatu kondisi yang disyaratkan dipenuhi. Pada proses seperti ini simbol flowchart Decision harus digunakan. Simbol decision akan berisi pernyataan yang akan diuji kebenarannya. Nilai hasil pengujian akan menentukan cabang mana yang akan ditempuh.

3. Struktur Algoritma Perulangan / Pengulangan



Gambar 6. Algoritma Perulangan

Dalam banyak kasus seringkali kita dihadapkan pada sejumlah pekerjaan yang harus diulang berkali.Salah satu contoh yang gampang kita jumpai adalah balapan mobil seperti tampak pada gambar ini Mobil-mobil peserta harus mengelilingi lintasan sirkuit berkali-kali sesuai yang ditetapkan dalam aturan lomba. Siapa yang mencapai garis akhir paling cepat, dialah yang menang.

Struktur pengulangan terdiri dari dua bagian :

1. Kondisi pengulangan, yaitu syarat yang harus dipenuhi untuk melaksanakan pengulangan. Syarat ini biasanya dinyatakan dalam ekspresi Boolean yang harus diuji apakah bernilai benar (true) atau salah (false)
2. Badan pengulangan (loop body), yaitu satu atau lebih instruksi yang akan diulang

   Pada struktur pengulangan, biasanya juga disertai bagian *inisialisasi* dan bagian *terminasi*. ***Inisialisasi*** adalah instruksi yang dilakukan sebelum pengulangan dilakukan pertama kali. Bagian insialisasi umumnya digunakan

untuk memberi nilai awal sebuah variable. Sedangkan ***terminasi*** adalah instruksi yang dilakukan setelah pengulangan selesai dilaksanakan. Ada beberapa bentuk pengulangan yang dapat digunakan, masing-masing dengan syarat dan karakteristik tersendiri. Beberapa bentuk dapat dipakai untuk kasus yang sama, namun ada bentuk yang hanya cocok untuk kasus tertentu saja. Pemilihan bentuk pengulangan untuk masalah tertentu dapat mempengaruhi kebenaran algoritma. Pemilihan bentuk pengulangan yang tepat bergantung pada masalah yang akan diprogram.

a. Struktur pengulangan dengan For

Pengulangan dengan menggunakan For, merupakan salah teknik pengulangan yang paling tua dalam bahasa pemrograman. Hampir semua bahasa pemrograman menyediakan metode ini, meskipun sintaksnya mungkin berbeda. Pada struktur For kita harus tahu terlebih dahulu seberapa banyak badan loop akan diulang. Struktur ini menggunakan sebuah variable yang biasa disebut sebagai loop's counter, yang nilainya akan naik atau turun selama proses pengulangan.

Contoh :

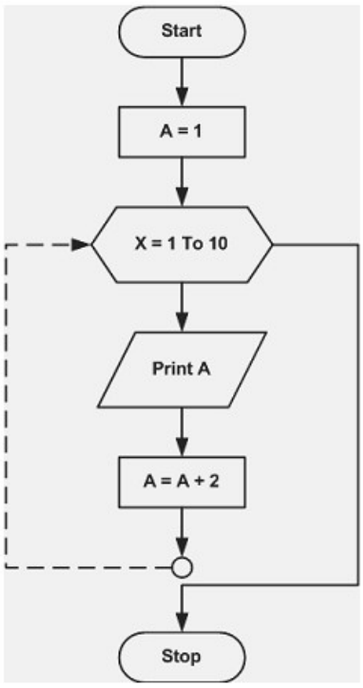
Diketahui sebuah himpunan A yang beranggotakan bilangan 1, 3, 5, .., 19. Buatlah flowchart untuk mencetak anggota himpunan tersebut.

Penyelesaian:

Pada contoh ini, kita mencoba menentukan hasil dari sebuah flowchart . Bagaimana menurut kalian jawabannya? Marilah kita uraikan jalannya fowchart tersebut. Pada flowchart, setelah Start, kita meletakkan satu proses yang berisi pernyataan A = 1. Bagian inilah yang disebut inisialisasi . Kita memberi nilai awal untuk A = 1. Variabel counter-nya adalah X dengan nilai awal 1 dan nilai akhir 10, tanpa increment (atau secara default increment-nya adalah 1). Ketika masuk ke badan loop untuk pertama kali maka akan dicetak langsung nilai variabel A. Nilai variabel A masih sama dengan 1. Kemudian proses berikutnya adalah pernyataan A = A + 2. Pernyataan ini mungkin agak aneh, tapi ini adalah sesuatu yang pemrograman. Arti dari pernyataan ini adalah gantilah

nilai A yang lama dengan hasil penjumlah nilai A lama ditambah 2. Sehingga A akan bernilai 3. Kemudian dilakukan pengulangan yang ke-dua. Pada kondisi ini nilai A adalah 3, sehingga yang tercetak oleh perintah print adalah 3. Baru kemudian nilai A kita ganti dengan penjumlahan A + 2. Nilai A baru

adalah 5. Demikian seterusnya. Sehingga output dari flowchart ini adalah 1, 3, 5, 7, .., 19.



b. Struktur pengulangan dengan While

Pada pengulangan dengan For, banyaknya pengulangan diketahui dengan pasti karena nilai awal (start) dan nilai akhir (end) sudah ditentukan diawal pengulangan. Bagaimana jika kita tidak tahu pasti harus berapa kali mengulang? Pengulangan dengan While merupakan jawaban dari permasalahan ini. Seperti halnya For, struktur pengulangan dengan While juga merupakan struktur yang didukung oleh hampir semua bahasa pemrograman namun dengan sintaks yang berbeda.

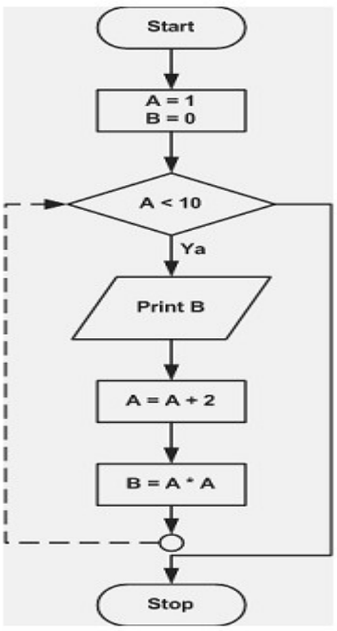
Struktur While akan mengulang pernyataan pada badan loop sepanjang kodisi pada While bernilai benar. Dalam artian kita tidak perlu tahu pasti berapa kali diulang. Yang penting sepanjang kondisi pada While dipenuhi maka pernyataan pada badan loop akan diulang.

Penyelesaian: Perhatikan Gambar 5.20. bisakah kalian menentukan hasil dari flowchart tersebut? Perhatikan tahapan eksekusi flowchart berikut ini.

1. Pada flowchart ini ada dua variabel yang kita gunakan yaitu A dan B. Kedua variabel tersebut kita inisialisasi nilai awalnya (A = 1 dan B = 0) sebelum proses loop terjadi. Variabel A adalah variabel counter.
2. Pada simbol decision, nilai A akan diperiksa apakah memenuhi kondisi (< 10). Jika Ya maka perintah berikutnya dieksekusi, jika tidak maka program akan berhenti. Pada awal eksekusi ini kondisi akan terpenuhi karena nilai A= 1.
3. Jalankan perintah Print B.

4. Nilai variabel A kemudian diganti dengan nilai A lama (1) ditambah 2. Sehingga nilai variabel A baru adalah 3. Sedangkan nilai variabel B = 9 (hasil perkalian A = 3).

5. Program akan berputar kembali untuk memeriksa apakah nilai variabel A masih lebih kecil dari 10. Pada kondisi ini nilai A = 3, sehingga kondisi masih terpenuhi. Kemudian langkah berulang ke langkah ke 3. Begitu seterusnya sampai nilai variabel A tidak lagi memenuhi syarat kurang dari 10.



Latihan Soal.

1. Buatlah Flowchart tentang menentukan apakah suatu bilangan adalah bilangan ganjil atau bilangan genap?
2. Buatlah Algoritma untuk Sebuah aturan menonton sebuah film tertentu adalah sebagai berikut, jika usia penonton lebih dari 17 tahun maka penonton diperbolehkan dan apabila kurang dari 17 tahun maka penonton tidak diperbolehkan nonton. Buatlah flowchart untuk permasalahan tersebut.

BAB III
Menjelaskan Data Flow Diagram (DFD)

A. KONSEP PERANCANGAN TERSTRUKTUR

Pendekatan perancangan terstruktur dimulai dari awal 1970. Pendekatan terstruktur dilengkapi dengan alat-alat (tools) dan teknik-teknik (techniques) yang dibutuhkan dalam pengembangan sistem, sehingga hasil akhir dari sistem yang dikembangkan akan diperoleh sistem yang strukturnya didefinisikan dengan baik dan jelas.

Melalui pendekatan terstruktur, permasalahan yang komplek di organisasi dapat dipecahkan dan hasil dari sistem akam mudah untuk dipelihara, fleksibel, lebih memuaskan pemakainya, mempunyai dokumentasi yang baik, tepat waktu, sesuai dengan anggaran biaya pengembangan, dapat meningkatkan produktivitas dan kualitasnya akan
lebih baik (bebas kesalahan)

B. DATA FLOW DIAGRAM (DFD)

Penggunaan DFD Sebagai Modeling Tool dipopulerkan Oleh Demacro & Yordan (1979) dan Gane & Sarson (1979) dengan menggunakan pendekatan Metoda Analisis Sistem Terstruktur.

Data Flow Diagram (DFD) adalah alat pembuatan model yang memungkinkan profesional sistem untuk menggambarkan sistem sebagai suatu jaringan proses fungsional yang dihubungkan satu sama lain dengan alur data, baik secara manual maupun komputerisasi. DFD ini sering disebut juga dengan nama Bubble chart, Bubble diagram, model proses, diagram alur kerja, atau model fungsi.

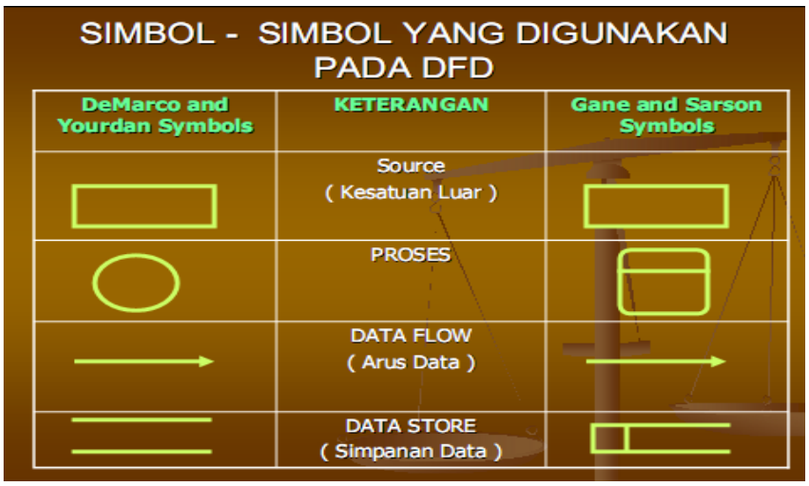
DFD ini adalah salah satu alat pembuatan model yang sering digunakan, khususnya bila fungsi-fungsi sistem merupakan bagian yang lebih penting dan kompleks dari pada data yang dimanipulasi oleh sistem.Dengan kata lain, DFD adalah alat pembuatan model yang memberikan penekanan hanya pada fungsi sistem.

DFD ini merupakan alat perancangan sistem yang berorientasi pada alur data dengan konsep dekomposisi dapat digunakan untuk penggambaran analisa maupun rancangan sistem yang mudah dikomunikasikan oleh profesional sistem kepada pemakai maupun pembuat program.

DFD merupakan alat yang digunakan pada metodologi pengembangan sistem yang terstruktur.

Kelebihan utama pendekatan aliran data, yaitu :

1. Kebebasan dari menjalankan implementasi teknis sistem.
2. Pemahaman lebih lanjut / jauh mengenai keterkaitan satu sama lain dalam sistem dan subsistem.
3. Mengkomunikasikan pengetahuan sistem yang ada dengan pengguna melalui diagram aliran data.
4. Menganalisis sistem yang diajukan untuk menentukan apakah data-data dan proses yang diperlukan sudah ditetapkan.
5. Dapat digunakan sebagai latihan yang bermanfaat bagi penganalisis, sehingga bisa memahami dengan lebih baik keterkaitan satu sama lain dalam sistem dan subsistem.
6. Membedakan sistem dari lingkungannya dengan menempatkan batas-batasnya.
7. Dapat digunakan sebagai suatu perangkat untuk berinteraksi dengan pengguna.
8. Memungkinkan penganalisis menggambarkan setiap komponen yang digunakan dalam diagram.



Gambar 7. Simbol DFD

keterangan:

1. Kesatuan luar / terminator :

   Entitas Internal / kesatuan diluar sistem yang akan memberikan input atau menerima output dari sistem, dapat berupa orang organisasi, sumber informasi lain atau penerima akhir dari suatu laporan

   contoh: :mahasiswa, siswa, dinas, yayasan, kepala sekolah, pegawai, direktur dll.

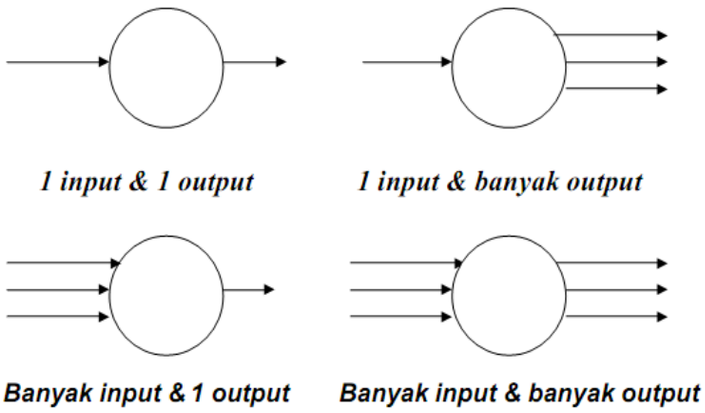
2. Proses :

   Merupakan kegiatan atau pekerjaan yang dilakukan oleh orang atau mesin komputer dimana aliran data masuk ditranformasikan ke aliran data keluar.

Contoh :

, dll

Ada empat kemungkinan yang dapat terjadi dalam proses sehubungan dengan input dan output :

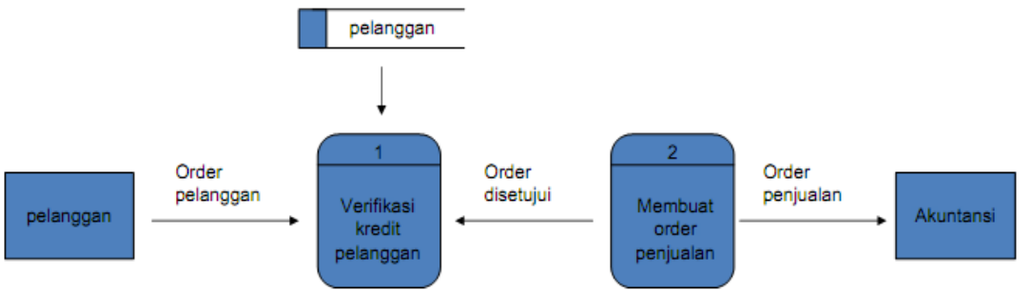


Gambar 8. Proses Input Output

Ada beberapa hal yang perlu diperhatikan tentang proses :

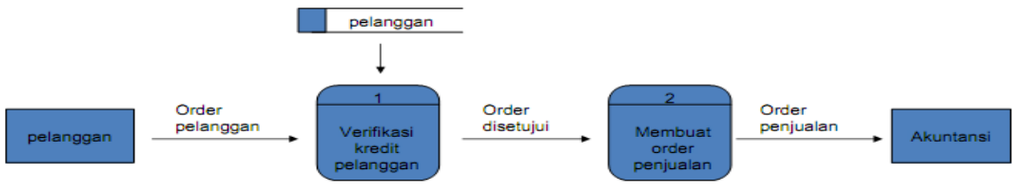
1) Proses harus memiliki input dan output.
2) Proses dapat dihubungkan dengan komponen terminator, data store atau proses melalui alur data.
3) Sistem/bagian/divisi/departemen yang sedang dianalisis oleh profesional sistem digambarkan dengan komponen proses.

Berikut ini merupakan suatu contoh proses yang salah :



Gambar 9. Contoh Proses Yang Salah

Pembuatan proses yang tepat :



Gambar 10. Contoh Proses Yang Tepat

3. Arus data :

   disimbolkan dengan anak panah, dimana arus data mengalir diantara proses , simpangan data, kesatuan luar, kesatuan ruang.

   Arus data dapat berbentuk sebagai berikut :

   „ „ Formulir atau dokumen yang digunakan perusahaan

   „ „ Laporan tercetak yang dihasilkan sistem

   „ „ Output dilayar komputer

   „ „ Masukan untuk komputer

   „ „ Komunikasi ucapan

   „ „ Surat atau memo

   „ „ Data yang dibaca atau direkam di file

   „ „ Suatu isian yang dicatat pada buku agenda

   „ „ Transmisi data dari suatu komputer ke komputer lain

4. Data Store :

   Data Store digunakan untuk membuat model sekumpulan paket data dan diberi nama dengan kata benda jamak.

   Data store ini biasanya berkaitan dengan penyimpanan-penyimpanan, seperti file atau database yang berkaitan dengan penyimpanan secara komputerisasi, misalnya file disket, file harddisk, file pita magnetik. Data store juga berkaitan dengan penyimpanan secara manual seperti buku alamat, file folder, dan agenda.

Aturan dalam DFD

Dalam penggambaran DFD, ada beberapa peraturan yang harus diperhatikan sehingga dalam penggambarannya tidak terjadi kesalahan, aturan tersebut yaitu :

a. Antar entitas tidak diijinkan terjadi hubungan atau relasi.
b. Tidak boleh ada aliran data antara entitas eksternal dengan data store.
c. Untuk alasan kerapian (menghindari aliran data yang bersilangan), entitas eksternal atau data store boleh digambar beberapa kali dengan tanda khusus, misalnya diberi nomor.

d. Satu aliran data boleh mengalirkan beberapa paket data.
e. Bentuk anak panah aliran data boleh bervariasi.
f. Semua objek harus mempunyai nama.
g. Aliran data selalu diawali atau diakhir dengan proses.
h. Semua aliran data harus mempunyai tanda arah.
i. Jumlah proses tidak lebih dari sembilan proses dalam sistem, jika melebihi maka sebaiknya dikelompokkan beberapa proses yang bekerja bersama-sama didalam suatu subsistem.

C. PEMBUATAN DFD

Tidak ada aturan baku untuk menggambarkan DFD. Tapi dari berbagai referensi yang ada, secara garis besar langkah untuk membuat DFD adalah :

1. Identifikasi terlebih dahulu semua entitas luar yang terlibat di sistem.
2. Identifikasi semua input dan output yang terlibat dengan entitas luar.
3. Buat Diagram Konteks (diagram context) Diagram ini adalah diagram level tertinggi dari DFD yang menggambarkan hubungan sistem dengan lingkungan luarnya.

   Caranya :

   a. Tentukan nama sistemnya.
   b. Tentukan batasan sistemnya.
   c. Tentukan terminator apa saja yang ada dalam sistem.
   d. Tentukan apa yang diterima/diberikan terminator dari/ke sistem.
   e. Gambarkan diagram konteks.
4. Buat Diagram Level Zero

   Diagram ini adalah dekomposisi dari diagram konteks. Caranya :

   a. Tentukan proses utama yang ada pada sistem.
   b. Tentukan apa yang diberikan/diterima masing-masing proses ke/dari sistem sambil memperhatikan konsep keseimbangan (alur data yang keluar/masuk dari suatu level harus sama dengan alur data yang masuk/keluar pada level berikutnya).
   c. Apabila diperlukan, munculkan data store (master) sebagai sumber maupun tujuan alur data.
   d. Gambarkan diagram level zero.
   e. Hindari perpotongan arus data
   f. Beri nomor pada proses utama (nomor tidak menunjukkan urutan proses).
5. Buat Diagram Level Satu

   Diagram ini merupakan dekomposisi dari diagram level zero.

   Caranya :

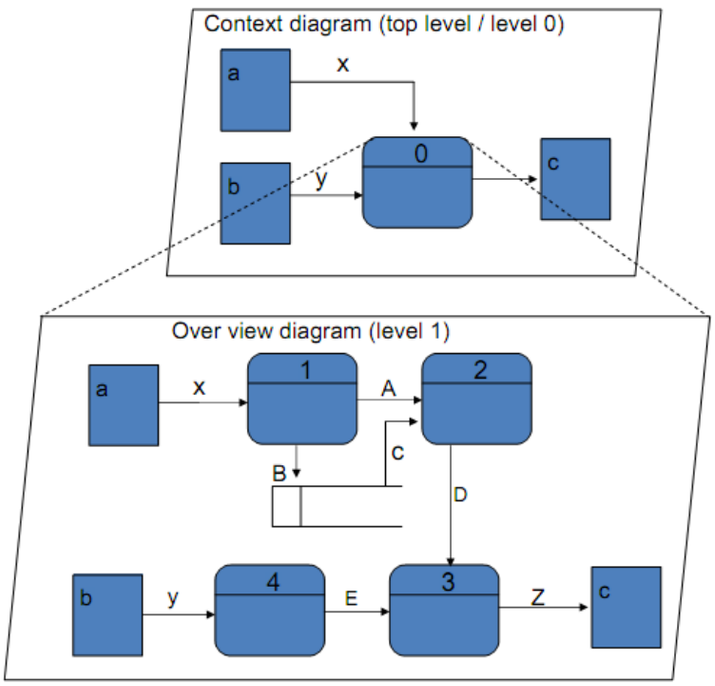
a. Tentukan proses yang lebih kecil (sub-proses) dari proses utama yang ada di level zero.
b. Tentukan apa yang diberikan/diterima masing-masing sub-proses ke/dari sistem dan perhatikan konsep keseimbangan.
c. Apabila diperlukan, munculkan data store (transaksi) sebagai sumber maupun tujuan alur data.
d. Gambarkan DFD level Satu
e. Hindari perpotongan arus data.
f. Beri nomor pada masing-masing sub-proses yang menunjukkan dekomposisi dari proses sebelumnya.
   Contoh : 1.1, 1.2, 2.1

6. DFD Level Dua, Tiga, …

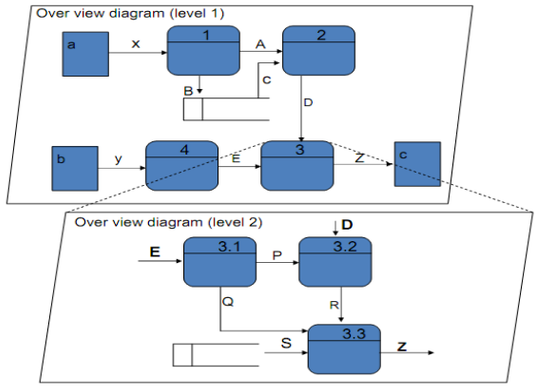
   Diagram ini merupakan dekomposisi dari level sebelumnya. Proses dekomposisi dilakukan sampai dengan proses siap dituangkan ke dalam program. Aturan yang digunakan sama dengan level satu.

Context diagram adalah DFD ruang lingkup dari sistem yang menunjukkan batas-batas sistem, external entitiy yang berinteraksi dengan sistem dan aliran data utama antara external entity dengan sistem. Context diagram menggambarkan keseluruhan sistem dalam suatu proses tunggal.

Contoh pembuatan DFD :



Gambar 11. Pembuatan DFD



Gambar 12. Pembuatan DFD 1

Contoh latihan pembuatan DFD :

Contoh Kasus 1

Pimpinan POLDA XYZ ingin mengetahui dengan pasti keadaan personilnya (riwayat personil, penugasan, prestasi, penghargaan, dll) secara periodik.

Bantulah keinginan Kapolda tersebut dalam bentuk perancangan sistem dengan menggunakan DFD !

Jawab Kasus 1

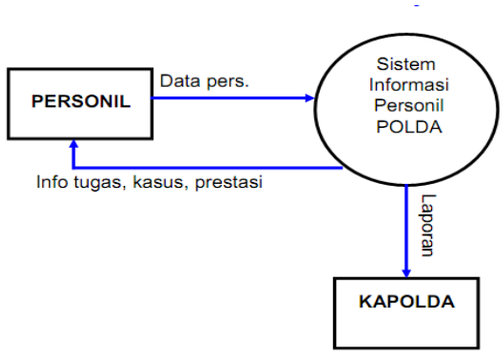
• Langkah pertama kita buat Diagram Konteks.

• Diidentifikasi entitas eksternal yang terlibat dalam sistem, yaitu:

• Personil

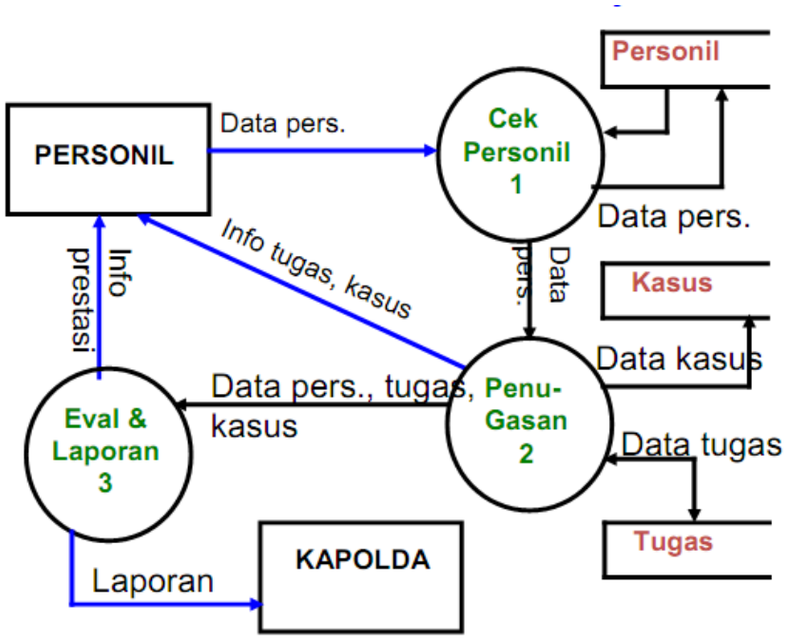
• Pimpinan (Kapolda)

• Sehingga diperoleh Diagram Konteks yaitu :



Gambar 13. Diagram Konteks

DFD Level 0 :



Gambar 14. DFD Level 0

BAB IV
Varian & Invariant

A. Pengertian Varian dan Invariant

Varian adalah jenis/turunan dari alur logikal yang digunakan untuk pra-built aplikasi sedangkan invarian adalah bentuk logical dari external algoritma (fungsi) yang terintegrasi dengannya. Dalam artian lain, varian adalah variabel yang tidak mempunyai tipe data dan invarian adalah variabel yang mempunyai tipe data.

B. Variabel, Tipe Data Dan Konstanta

Variabel, konstanta dan tipe data merupakan tiga hal yang akan selalu kita jumpai ketika kita membuat program. Bahasa pemrograman apapun dari yang paling sederhana sampai yang paling kompleks, mengharuskan kita untuk mengerti ketiga hal tersebut.

1. **Variabel** adalah tempat dimana kita dapat mengisi atau mengosongkan nilainya dan memanggil kembali apabila dibutuhkan. Setiap variabel akan mempunyai nama (identifier) dan nilai.

   Mendeklarasikan Variabel

   Variable digunakan untuk menyimpan data. Dengan demikian tanpa variabel, sebuah program tidak akan dapat melakukan apa-apa. pendeklarasian variable bersifat opsional, maksudnya Anda bisa mendeklarasikannya atau tidak. Bila Anda memberi nilai pada suatu variable, Anda telah mendeklarasikan variable tersebut. Sebagai
   contoh :
   X = 50; // X bernilai 50

   bahwa telah dideklarasikan variable X. Cara seperti ini merupakan pendeklarasian variabel secara implisit. Untuk mendeklarasikan variabel secara eksplisit, tulislah nama variable didahului dengan kata kunci var, contoh :
   var X; // mendeklarasikan sebuah variable dengan nama X
   untuk mendeklarasikan beberapa variabel dalam satu baris, Anda cukup menuliskannya dengan tanda pemisah koma, contoh :
   var X, Y, Z;
   // mendeklarasikan 3 buah variable, yaitu X, Y dan Z

Dalam PHP setiap nama variable diawali tanda dollar ($). Misalnya nama variable a dalam PHP ditulis dengan $a. Jenis suatu variable ditentukan pada saat jalannya program dan tergantung pada konteks yang digunakan.

Untuk dapat menggunakan variabel, ada dua langkah yang harus dilakukan, deklarasi dan inisialisasi.

Inisialisasi variabel

Inisialisasi variabel adalah mengisi nilai untuk pertama kalinya ke dalam variabel.

Contoh inisialisasi :

```
$namaDepan = “Endy”;
$namaBelakang = “Muhardin”;
$jumlahBarang = 3;
$harga = 1000;
$nama = 'LUG STIKOMP Surabaya';
$angka_1 = 1;
$angka_2 = 2;
$hasil = $angka_1 + $angka_2;
```

Meskipun pendeklarasian variabel bersifat opsional, tapi sebelum diberi nilai, variabel tidak dapat digunakan.

Pada saat dideklarasikan suatu variabel dapat langsung diinisialisasi atau diberi nilai tertentu, contoh:

```
var X = “hallo”;
var Y = 50;
```

Pemberian nama variabel harus mengikuti aturan yang ditetapkan oleh bahasa pemrograman yang kita gunakan. Namun secara umum ada aturan yang berlaku untuk hampir semua bahasa pemrograman.

Aturan-aturan tersebut yaitu:

a) Nama variabel harus diawali dengan huruf.
b) Tidak boleh menggunakan spasi pada satu nama variabel. Spasi bisa diganti dengan karakter underscore (_).
c) Nama variabel tidak boleh mengandung karakter-karakter khusus, seperti : .,+, -, *, /, <, >, &, (,) dan lain-lain.
d) Nama variabel tidak boleh menggunakan kata-kata kunci d bahasa Pemrograman

Penamaan yang benar	Penamaan yang salah
`namasiswa`	`nama siswa` (salah karena menggunakan spasi)
`XY12`	`12X` (salah karena dimulai dengan angka)
`harga_total`	`harga.total` (salah karena menggunakan karakter .)
`JenisMotor`	`Jenis Motor` (salah karena menggunakan spasi)
`alamatRumah`	`for` (salah karena menggunakan kata kunci bahasa pemrograman)

contoh variabel1.php:

```
<?php
$a="14";
$b="20";
$hasil=$a+$b;
echo($hasil);
?>
```

contoh variabel2.php:

```
<?php
$nim = "0411500400";
$nama = 'Chotimatul Musyarofah';
echo "NIM  : " . $nim . "<br>";
echo "Nama : $nama";
?>
```

2. Tipe Data

Tipe data adalah jenis data yang dapat diolah oleh komputer untuk memenuhi kebutuhan dalam pemrograman komputer.

Pada PHP, tipe data variabel tidak didefinisikan oleh programmer, akan tetapi secara otomatis ditentukan oleh intepreter PHP.

Berikut ini adalah beberapa tipe data yang didukung oleh PHP

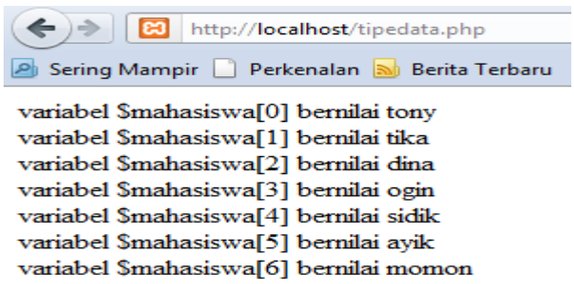
Macam-macam tipe Data :

TipeData	Keterangan
Integer	Digunakan untuk semua angka
String	Digunakan untuk semua huruf, angka, spasi, dan simbol
Double	Digunakan untuk bilangan real
Boolean	Digunakan untuk nilai True atau False
Array	Digunakan untuk menampung beberapa data sekaligus
Object	Digunakan untuk class

Latihan tipe data

```
<?php
$mahasiswa[0] = 'tony';
$mahasiswa[1] = 'tika';
$mahasiswa[2] = 'dina';
$mahasiswa[3] = 'ogin';
$mahasiswa[4] = 'sidik';
$mahasiswa[5] = 'ayik';
$mahasiswa[6] = 'momon';
$mahasiswa[7] = 'ippin';
for ($i=0; $i<7; $i++) {
  echo "variabel \$mahasiswa[$i] bernilai $mahasiswa[$i] <br/>";
}
?>
```

Hasil :



3. Konstanta

Konstanta adalah variabel yang nilai datanya bersifat tetap dan tidak bisa diubah. Untuk mendefinisikan konstanta dalam PHP, menggunakan fungsi define()

Hampir sama dengan variabel, konstanta juga digunakan untuk penyimpanan nilai sementara. Namun perbedaan konstanta dengan variabel adalah pada konstanta an da tidak dapat mengubah

nilainya jika sudah dideklarasikan. Cara pendeklarasiannya pun berbeda dengan v ariabel. Pada konstanta digunakan keyword define untuk mendeklarasikan variabe l. Konstanta juga tidak diawali dengan tanda $ (dollar).

```
define('HARGA', 1500);
define('NAMA', 'LUG STIKOMP Surabaya');
```

Dapat dilihat pada kode diatas bahwa kita selalu gunakan huruf KAPITAL untuk konstanta. Hal ini tidak harus dilakukan namun semacam peraturan tidak tertulis ji ka konstanta maka sebaiknya gunakan huruf kapital.

Contoh konstanta.

```
<?
define ("NAMA", "Achmad Solichin");
define ("NILAI", 90);
```

```
//NAMA = "Muhammad"; //akan menyebabkan error
echo "Nama : " . NAMA;
echo "<br>Nilai : " . NILAI;
?>
```

Hasil :

Nama : Achmad Solichin

Nilai : 90

Latihan konstanta

```
<?php
$menu1 ="Menu : Nasi Pecel";
define ('HARGA1',2000);
$menu ="Menu : Nasi Rawon";
define ('HARGA',5000);
echo"$menu1";
echo"<br>";
echo"Harga : ";
echo(HARGA);
?>
```

Hasil :

Menu : Nasi Rawon
Harga : 2000

4. Operator

Operator adalah simbol yang digunakan dalam program untuk melakukan suatu operasi. Operator berguna untuk melakukan suatu operasi pada suatu nilai. Operator di PHP sangatlah umum sehingga mudah untuk dipahami. Disini kita akan membahas operator yang sering digunakan.

Jenis Operator	Operator	Contoh	Keterangan
Aritmatika	+	$a + $b	Pertambahan
	-	$a - $b	Pengurangan
	*	$a * $b	Perkalian
	/	$a / $b	Pembagian
	%	$a % $b	Modulus, sisa pembagian
Penugasan	=	$a = 4;	$a diisi dengan 4
Bitwise	&	$a & $b	Bitwise AND
	|	$a | $b	Bitwise OR
	^	$a ^ $b	Bitwise XOR
	~	~$b	Bitwise NOT
	<<	$a << $b	Shift Left
	>>	$a >> $b	Shift Right
Perbandingan	==	$a == $b	Sama dengan
	===	$a === $b	Identik
	!=	$a != $b	Tidak sama dengan
	<>	$a <> $b	Tidak sama dengan
	!==	$a !== $b	Tidak identik
	<	$a < $b	Kurang dari
	>	$a > $b	Lebih dari
	<=	$a <= $b	Kurang dari sama dengan
	>=	$a >= $b	Lebih dari sama dengan
Logika	and	$a and $b	TRUE jika $a dan $b TRUE
	&&	$a && $b	TRUE jika $a dan $b TRUE
	or	$a or $b	TRUE jika $a atau $b TRUE
	||	$a || $b	TRUE jika $a dan/atau $b TRUE
	xor	$a xor $b	TRUE jika $a atau $b TRUE, tapi tidak keduanya
	!	!$a	TRUE jika $a FALSE
String	.	$a . $b	Penggabungan string $a dan $b

Latihan(Operator)

```
<?php
$gaji = 1000000;
$pajak = 0.1;
$hasil = $gaji - ($gaji*$pajak);
echo "Gaji sebelum pajak = Rp. $gaji <br>";
echo "Gaji yang dibawa pulang = Rp. $hasil";
?>
```

Hasil :

Gaji sebelum pajak = Rp. 1000000
Gaji yang dibawa pulang (pajak 10 %)= Rp. 900000

Latihan Soal :

1. Tentukan salah atau benar pada nama-nama variabel berikut ini. Jika salah cobalah berikan alasan.

   a. nama.guru

   b. NamaGuru

   c. 2x

   d. harga/buku

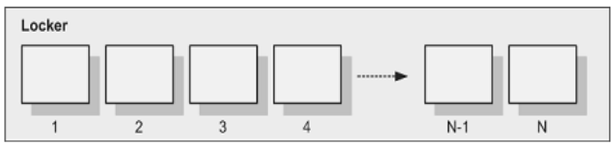
   e. hargaPerBuku

2. Tentukan tipe data yang cocok untuk hal-hal berikut ini (perhatikan ini bukan nama variabel) dan jelaskan alasannya.
   a. Jumlah murid
   b. Berat badan
   c. Tinggi badan
   d. Nama siswa
   e. Tempat lahir
   f. Tanggal lahir

BAB V
ARRAY

A. Definisi Array

Variabel-variabel yang kita gunakan selama ini adalah variable biasa yang memiliki sifat bahwa sebuah nama variable hanya dapat menyatakan sebuah nilai numeric atau string pada suatu saat. Apabila kita ingin memberi nilai yang baru pada variable tersebut maka nilai lama akan hilang tergantikan oleh nilai yang baru. Bagaimana apabila kita ingin menyimpan beberapa nilai/data dalam sebuah variable dengan nama yang sama, tetapi semua nilai tetap tersimpan? Solusi yang dapat dilakukan adalah dengan menggunakan indeks pada nama variable tersebut. Cara ini biasa disebut dengan array. Array adalah struktur data yang menyimpan sekumpulan elemen yang bertipe sama, setiap elemen diakses langsung melalui indeksnya. Array merupakan tipe data terstruktur yang berguna untuk menyimpan sejumlah data yang bertipe sama. Bagian yang menyusun array disebut elemen array, yang masing-masing elemen dapat diakses tersendiri melalui indeks array. Indeks array haruslah tipe data yang menyatakan keterurutan, misalnya integer atau string. Array dapat dianalogikan sebagai sebuah lemari atau locker yang memiliki sederetan kotak penyimpanan yang diberi nomor berurutan (lihat Gambar 8). Untuk menyimpan atau mengambil sesuatu dari kotak tertentu kita hanya cukup mengetahui nomor kotaknya saja.



Gambar 15. Lemari dengan banyak kotak laci di dalamnya

Pada variabel array, kita tidak hanya menentukan tipe datanya saja, tetapi juga jumlah elemen dari array tersebut atau dalam hal ini adalah batas atas indeksnya. Pada banyak bahasa pemrograman seperti *PHP, C++, Visual Basic,* dan beberapa yang lainnya, nilai indeks awal adalah 0 bukan 1. Cara menuliskan variabel array berbeda- beda tergantung bahasa pemrograman apa yang dipakai. Tetapi yang pasti tipe data harus disebutkan dan batas atas indeks harus ditentukan. Untuk mengisi data pada array kita dapat langsung menentukan pada indeks berapa kita akan isikan demikian juga untuk memanggil atau menampilkan data dari array.

“Elektronika”
“Telekomunikasi”
“Elektro Industri”
“Teknologi Informasi”
“Teknik Kimia”
Elemen –
Elemen Array
Array Jurusan

Dalam terminology array, array jurusan diatas bias dikatakan mempunyai 5 buah elemen. Setiap elemen mempunyai sebuah nilai. Elemen pertama berisi string “Elektronika”,
elemen kedua berisi string “Telekomunikasi”, dan seterusnya.

Membuat Array
Suatu array dapat dibuat dengan menggunakan konstruksi array. Sebagai contoh array jurusan diatas dapat dibentuk dengan menggunakan pernyataan sebagai berikut ini :

```
$jurusan = array (“Elektronika”,
 “Telekomunikasi”,
  “Elektro Industri”,
  “Teknologi Informasi”,
“Teknik Kimia”);
```

Cara yang lain, anda bisa menggunakan cara sebagai berikut :

```
$jurusan[] = “Elektronika”
$jurusan[] = “Telekomunikasi”
$jurusan[] = “Elektro Industri”
$jurusan[] = “Teknologi Informasi”
$jurusan[] = “Teknik Kimia”
```

Angka yang diletakkan di dalam tanda [] biasanya disebut kunci atau indeks. PHP, secara bawaan menggunakan indeks dimulai dengan nol.

“Elektronika”

“Telekomunikasi”

“Elektro Industri”

“Teknologi Informasi”

“Teknik Kimia”

Dalam prakteknya, indeks tidak harus dimulai dari nol. Bahkan anda bias menciptakan indeks yang tidak berurut.

```
$bilangan[7] = 100;
$bilangan[13] = 150;
$bilangan[20] = 45;
```

Tampak pada indeks yang digunakan dimulai dari 7, dan berikutnya tidak menggunkaan indeks 8 dan 9, melainkan 13 dan 20. hal ini boleh – boleh saja.

Perlu diketahui, bila anda menuliskan pernyataan seperti :

```
$bilangan[7] = 100;
$bilangan[13] = 150;
$bilangan[20] = 45;
$bilangan[] = 57;
```

Maka angka 21 akan disimpan ke elemen array yang memiliki indeks berupa 21 (20 + 1).

Mengambil isi Array

Untuk mengambil isi array, anda bias menggunakan notasi :

```
 $nama_array[indeks]
```

Contoh :

```
 Print ($jurusan[0]);
```

Akan menampilkan isi elemen pertama array jurusan.

Mengetahui Jumlah elemen Array

PHP menyediakan fungsi bernama count yang berguna untuk mendapatkan jumlah

elemen array. Fungsi ini memerlukan argument berupa array bersangkutan. Sebagai

contoh :

$jurusan[0]

Index

$jurusan[1]

$jurusan $jurusan[2]
$jurusan[3]
$jurusan[4] $musik = array (“Jazz”, “Blues”, “Fusion”);
Printf(“Jumlah elemen : %d”, count($musik));
Akan menampilkan :
Jumlah elemen = 3

Mengakses Elemen Array menggunakan kalang
Untuk menampilakan array yang berjumlah banyak, tidaklah praktis jika memakai
sederetan instruksi sebagai berikut :
*print ("nama_array[0]
]n");*
*print ("nama_array[1]
]n");*
...
*print ("nama_array[20]
]n");*

Cara yang lebih baik adalah dengan menggunakan fungsi for. Adapun perintahnya
sebagai berikut :

For ($i = 0; $i <= 20 ; $i++)
*Print(“nama_array[$i]
”);*

Array dengan Indeks berupa string
PHP memperkenankan indeks bertipe string. Sebagai contoh, anda bisa membuat array
seperti berikut :
$hari[“Sunday”] = “Minggu”;
$hari[“Monday”] = “Senin”;
$hari[“Tuesday”] = “Selasa”
$hari[“Wednesday”] = “Rabu”

Pada contoh diatas,
- *Elemen berindeks “Sunday” berisi string “Minggu”;*
- *Elemen berindeks “Sunday” berisi string “Senin”;*
- *Elemen berindeks “Tuesday” berisi string “Selasa”;*
- *Elemen berindeks “Wednesday” berisi string “Rabu”;*