Algoritma dan Pemrograman

Pokok Bahasan

- Algoritma
- Program dan Pemrograman
- Notasi Algoritmik
- Struktur Dasar Algoritma

Algoritma

Algoritma adalah...

- Urutan langkah-langkah untuk memecahkan suatu masalah
- Deretan langkah-langkah komputasi yang mentransformasikan data masukan menjadi keluaran
- Deretan instruksi yang jelas untuk memecahkan masalah, yaitu untuk memperoleh keluaran yang diinginkan dari suatu masukan dalam jumlah waktu yang terbatas
- Prosedur komputasi yang terdefinisi dengan baik yang menggunakan beberapa nilai sebagai masukan dan menghasilkan beberapa nilai yang disebut keluaran

Contoh Algoritma

- Masalah mengurutkan kartu
  - Kartu = 50 buah
  - Cari kartu dengan nomor terkecil terlebih dahulu
  - Letakkan pada posisi pertama (atas)
  - Cari kartu dengan nomor terkecil berikutnya
  - Letakkan kartu tersebut dibawah kartu pertama
  - Begitu seterusnya

Contoh Algoritma

- Terjadi pengulangan 2 langkah penting, yaitu:

  (i) cari kartu dengan nomor terkecil

  (ii) tempatkan kartu tersebut pada posisi yang tepat

- Secara garis besar dapat dituliskan langkah-langkah pengurutan n buah kartu :

  1. Cari kartu dengan nomor terkecil di antara kartu yang tersisa
  2. Tempatkan kartu tersebut pada posisi yang tepat
  3. Ulangi kembali dari langkah 1 sebanyak n-1 kali

- Langkah-langkah yang ditulis di atas itulah yang disebut **algoritma** (pada kasus ini disebut algoritma pengurutan)

Algoritma dan Proses

- Algoritma mengerjakan sebuah **proses**.
- Benda yang mengerjakan proses disebut pemroses (***processor***).
- Dapat berupa :
  - manusia
  - komputer
  - robot
  - alat elektronik
- Pemroses melakukan suatu proses dengan melaksanakan atau mengeksekusi algoritma yang menjabarkan proses tersebut.

Pemecah Masalah

- Setiap masalah mempunyai **algoritma pemecahannya**.
- Tugas kita sebagai pemecah masalah (***problem solver***) untuk mendeskripsikan langkah-langkah penyelesaiannya.

Contoh Algoritma Pemecahan

Misalkan ada dua buah ember :

Ember merah berisi air merah dan ember biru berisi air biru. Volume air di kedua ember sama.

Bagaimana mempertukarkan isi kedua ember itu sedemikian sehingga nantinya ember merah akan berisi air berwarna biru dan sebaliknya ember biru berisi air berwarna merah?

Contoh algoritma pemecahan

Perlu ember kosong tambahan sebagai tempat penampungan sementara agar air tidak tercampur.

Contoh algoritma pemecahan

Langkah-langkah pertukaran :

1.Tuangkan air dari ember merah ke ember hijau

2.Tuangkan air dari ember biru ke ember merah

3.Tuangkan air dari ember hijau ke ember biru

Keadaan akhir setelah pertukaran :

Ember merah berisi air biru, ember biru berisi air merah, ember hijau kosong

Latihan Soal

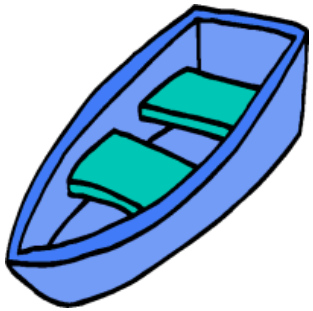
Latihan Soal 1: *Water Jug Problem*

- Misalkan Anda mempunyai dua buah ember, masing-masing bervolume 5 liter dan 3 liter.

- Anda diminta mendapatkan air dari sebuah danau sebanyak 4 liter dengan menggunakan bantuan kedua ember tersebut, terserah bagaimana caranya.

Latihan soal 2 : *River Problem*

- Misalkan seorang penggembala tiba di tepi sebuah sungai. penggembala tersebut membawa seekor kambing, seekor serigala, dan sekeranjang sayur.
- Mereka bermaksud hendak menyebrangi sungai.
- Penggembala itu menemukan sebuah perahu kecil di pinggir sungai tetapi sayang hanya dapat memuat 1 bawaan saja setiap kali menyebrang.



Latihan soal 2 : *River Problem*

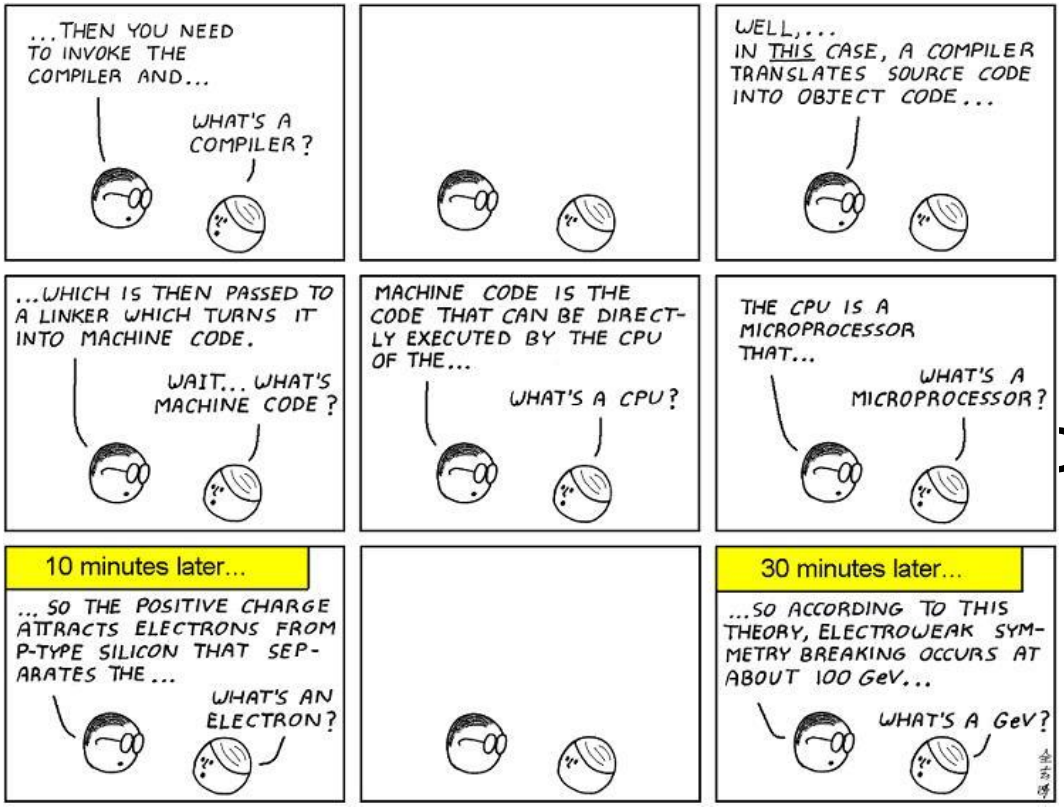
- Situasinya dipersulit dengan kenyataan bahwa serigala tidak dapat ditinggal berdua dengan kambing (karena serigala akan memangsa kambing) atau kambing tidak dapat ditinggal berdua saja dengan sekeranjang sayur (karena sayur akan dimakan kambing).
- Bagaimana algoritma si penggembala menyebrangkan seluruh bawaannya itu sehingga mereka sampai ke seberang sungai dengan selamat?
- Tentu saja hanya si penggembala yang bisa mendayung perahu.

Penyelesaian *Water jug*

Banyak kemungkinan penyelesaian, salah satunya :

ALGORITMA mendapatkan air dengan volume 4 liter :

1. Isi penuh ember 3L dengan air. {ember 3L berisi 3L}
2. Tuangkan air dari ember 3L ke dalam ember 5L. {ember 5L berisi 3L}
3. Isi penuh ember 3L dengan air. {ember 3L berisi 3L}
4. Tuangkan air dari ember 3L ke dalam ember 5L. {di ember 3L berisi air 1L}
5. Buang seluruh air dari ember 5L. {ember 5L kosong}
6. Tuangkan air dari ember 3L ke dalam ember 5L. {ember 5L berisi 1L}
7. Isi penuh ember 3L dengan air. {ember 3L berisi 3L}
8. Tuangkan air dari ember 3L ke dalam ember 5L. {ember 5L berisi 1L + 3L = 4L}



Pemrograman

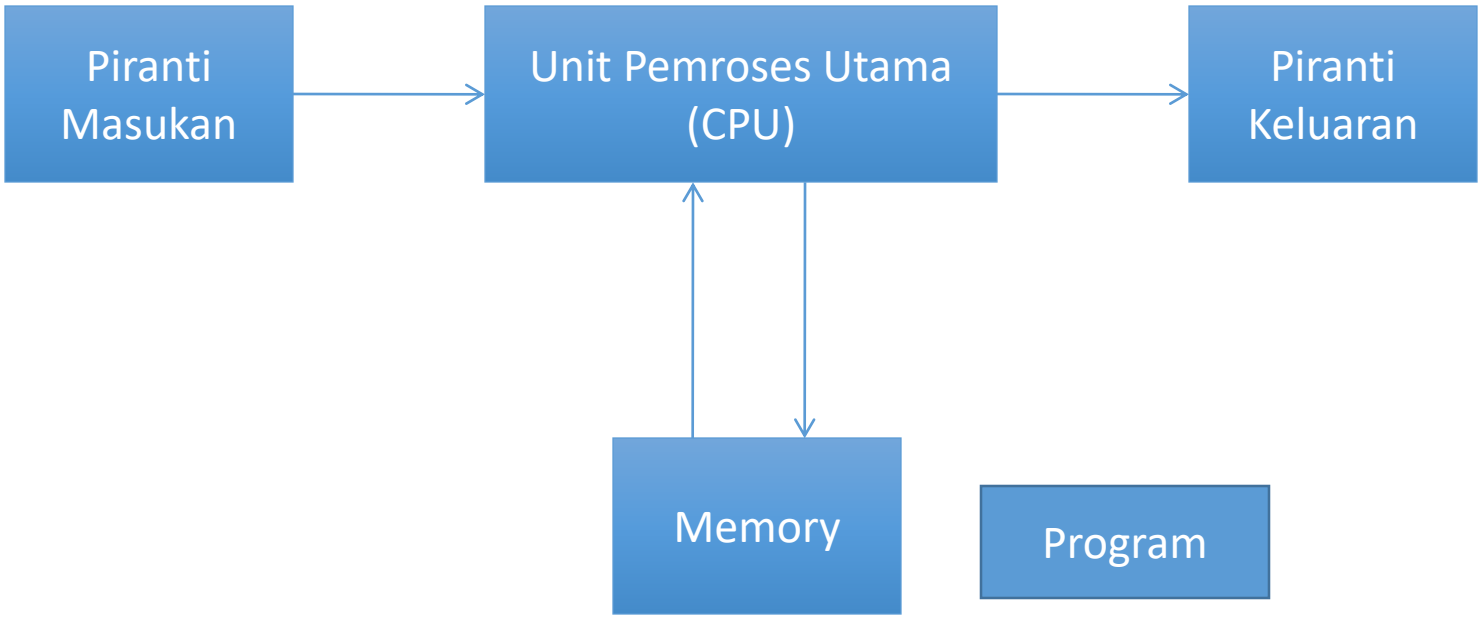
Program dan Pemrograman

- Algoritma baru efektif jika dijalankan oleh sebuah pemroses (processor) ➔ fokus ke komputer sebagai processor.
- Agar komputer mengerti perintah yang dimaksud dalam algoritma, maka perintah tersebut harus ditulis dalam bahasa khusus, yaitu bahasa komputer.
- Algoritma yang ditulis dalam bahasa komputer dinamakan **program**.
- Bahasa komputer yang digunakan dalam menulis program dinamakan **bahasa pemrograman.**
- Orang yang membuat program komputer disebut **pemrogram.**
- Kegiatan merancang dan menulis program disebut **pemrograman.**

Ciri-ciri Algoritma

- Harus berhenti setelah mengerjakan sejumlah langkah terbatas.
- Setiap langkah harus didefinisikan dengan tepat dan tidak berarti dua (ambigu).
- Memiliki nol atau lebih masukan (input). Masukan ialah besaran yang diberikan kepada algoritma untuk diproses.
- Mempunyai nol atau lebih keluaran (output). Keluaran dapat berupa pesan atau besaran yang memiliki hubungan dengan masukan.
- Harus efektif, sederhana sehingga dapat dikerjakan dlm sejumlah waktu yang masuk akal.

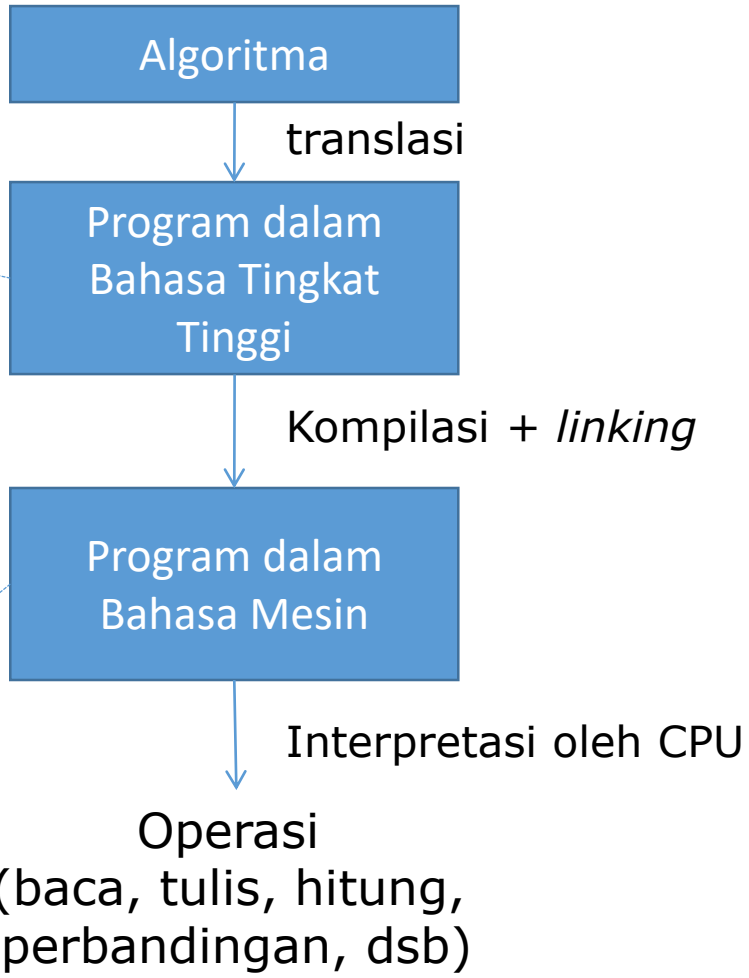
Mekanisme Kerja Komponen Komputer



Tingkatan Bahasa Pemrograman

Bahasanya lebih manusiawi, lebih dekat ke bahasa manusia (terutama bahasa Inggris). Program dengan bahasa ini tidak bisa langsung dieksekusi oleh komputer, tapi diterjemahkan terlebih dahulu oleh translator bahasa yang biasa disebut kompilator (compiler) ke dalam bahasa mesin sebelum akhirnya dieksekusi oleh CPU. Contoh : Pascal, Cobol, Basic, Fortran, C, C++, Java, dll.

Dirancang agar instruksinya langsung dikerjakan oleh komputer, tanpa harus melalui penerjemah (translator). Contoh : bahasa *assembly*. Sifatnya primitif, sederhana, dan relatif sulit dipahami manusia.



Interpreter

- Interpreter adalah suatu program komputer yang mampu menerjemahkan program dari bahasa tingkat tinggi yang dimengerti oleh manusia dan langsung menjalankan program tersebut. Kerja interpreter seperti penerjemah untuk turis yang langsung menerjemahkan kalimat demi kalimat yang dikatakan oleh sang turis.
- Setiap kali kita membutuhkan program tersebut, maka interpreter akan bekerja menerjemahkan program dari bahasa tingkat tinggi ke bahasa mesin untuk dieksekusi. Jadi siklus kerja ketika kita membuat program dengan interpreter adalah: tulis/edit program, eksekusi.

Kompilator

- Kompilator adalah suatu program komputer yang membaca seluruh program dari bahasa tingkat tinggi yang dimengerti oleh manusia dan kemudian menerjemahkan keseluruhan program tersebut dalam bahasa mesin.
- Program yang sudah diterjemahkan tersebut akhirnya akan dijalankan oleh komputer.
- Kerja kompilator seperti penerjemah buku yang akan menerjemahkan seluruh buku sekaligus, sehingga setiap orang bisa mengerti makna buku dalam bentuk terjemahannya.
- Kompilator hanya perlu bekerja sekali saja menerjemahkan bahasa tingkat tinggi ke bahasa mesin, dan jika kita membutuhkan kembali program tersebut, kita hanya perlu menjalankannya, kompilator tidak perlu bekerja lagi. Jadi siklus kerja jika kita memakai kompilator adalah: tulis/edit program, kompilasi, eksekusi

Debugger

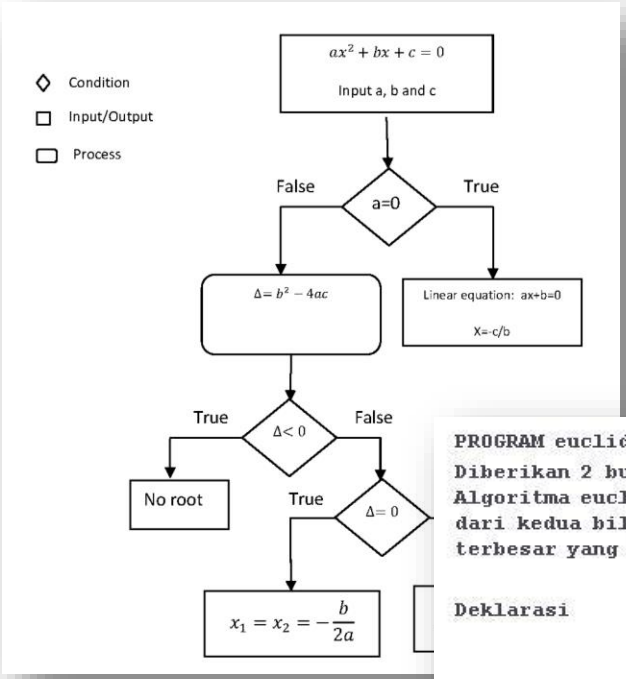
- Kesalahan pertama yang ditemukan pada salah satu komputer pertama (yang saat itu masih sangat besar) adalah karena adanya serangga/kutu (*bug*) yang menyebabkan komputer tidak bekerja. Sejak saat itu semua kesalahan, baik di bidang *hardware* maupun *software* komputer disebut dengan *bug* (istilah ini lebih umum di bidang *software* dibanding *hardware*).
- Proses untuk menemukan kesalahan program disebut juga dengan proses pencarian bug (istilah proses ini adalah *debug*). Dalam pencarian kesalahan ini terkadang diperlukan program pembantu yang dinamakan *debugger*. Program ini akan membantu pemrogram untuk melihat bagaimana eksekusi program dilakukan oleh komputer, dan melihat kesalahan yang mungkin ada ketika program sedang berjalan.

Editor dan Kompilator

- Untuk memasukkan program ke dalam komputer, kita perlu tools yang dinamakan editor. Editor adalah program yang mampu menerima teks dari manusia, dan menyimpannya ke dalam bentuk digital yang dimengerti komputer. Editor juga memungkinkan kita melakukan koreksi terhadap pengetikan yang kita lakukan (menghapus teks, menyalin teks, dan lain-lain). Bentuk kode program yang kita masukkan ini disebut dengan kode sumber atau *source code*.
- Untuk menjalankan program yang sudah kita ketikkan, kita akan membutuhkan kompilator atau interpreter. Pada bahasa Pascal, kompilator lebih umum dipakai dibanding interpreter. Perlu diperhatikan bahwa editor dan kompilator adalah dua program yang terpisah dan berbeda.

IDE

- Sebuah IDE (Integrated Development Environment) adalah program yang menggabungkan fungsi editor dan kompilator (serta terkadang debugger) dalam satu paket.
- IDE saat ini semakin populer, bahkan banyak orang yang menyangka bahwa IDE sama dengan kompilator.
- Sebuah IDE mungkin saja sekaligus memiliki fungsi kompilator, tapi tidak selalu demikian, terkadang IDE hanya menyediakan fungsi editor, dan akan memanggil kompilator yang sesungguhnya ketika kita akan mengkompilasi program.




```
PROGRAM euclidean
Diberikan 2 buah bilangan bulat tak negatif m dan n(m≥n).
Algoritma euclidean mencari pembagi bersama  terbesar, gcd,
dari kedua bilangan tersebut, yaitu bilangan bulat positif
terbesar yang habis membagi m dan n.

Deklarasi
            M, n : integer
            R    : integer

Algoritma :
Read(m,n) {m≥n}
While n ≠ 0 do
            R ← m MOD n
            M ← n
            N ← r
Endwhile
{Kondisi selesai pengulangan: n = 0, maka gcd(m,n) = m}
Write(m)
```

goritmik

Notasi Algoritmik

- Notasi algoritmik dibuat independen dari spesifikasi bahasa pemrograman dan perangkat keras komputer yang mengeksekusinya.
- Notasi algoritmik dapat diterjemahkan ke dalam berbagai bahasa pemrograman.
- Notasi algoritmik bukan bahasa pemrograman, sehingga siapapun dapat membuat notasi algoritmik yang berbeda.

Notasi Algoritmik

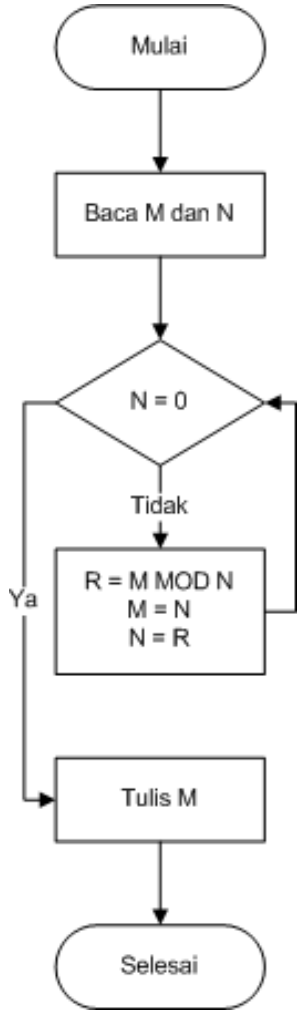
Notasi I : menyatakan langkah-langkah algoritma dengan untaian kalimat deskriptif.

```
PROGRAM Euclidean

Diberikan 2 buah bilangan bulat tak negatif m dan n    (m≥n).
Algoritma Euclidean mencari pembagi bersama      terbesar,
gcd, dari kedua bilangan tersebut, yaitu bilangan      bulat
positif terbesar yang habis membagi m dan n.

ALGORITMA

1. Jika n = 0 maka

    m adalah jawabannya;

    stop.

    tetapi jika n ≠ 0, lanjut ke langkah 2.

2. Bagilah m dengan n dan misalkan r adalah sisanya.

3. Ganti nilai m dengan n dan nilai n dengan nilai r, lalu
ulang kembali ke langkah 1.
```

Notasi Algoritmik

Notasi II : Menggunakan bagan alir (*flowchart*) sehingga aliran instruksi di dalam program digambarkan secara visual.

Cocok untuk program kecil.



Notasi Algoritmik

Notasi III :

Menggunakan pseudo-code, lebih praktis, merupakan campuran bahasa alami (bahasa manusia) dengan bahasa pemrograman. Namun tidak seperti bahasa pemrograman yang direpotkan dengan tanda titik koma, indeks, format keluaran, dll.

Notasi Algoritmik

```
PROGRAM Euclidean
Diberikan 2 buah bilangan bulat tak negatif m dan n(m≥n). Algoritma
Euclidean mencari pembagi bersama  terbesar, gcd, dari kedua
bilangan tersebut, yaitu bilangan bulat positif terbesar yang habis
membagi m dan n.

DEKLARASI
      m, n : integer
      r    : integer

ALGORITMA :
read(m,n) {m≥n}
while n ≠ 0 do
      r ← m MOD n
      m ← n
      n ← r
endwhile
{kondisi selesai pengulangan: n = 0, maka gcd(m,n) = m}
write(m)
```

Paradigma Pemrograman

.. cara pandang kita terhadap penyelesaian masalah pemrograman (atau sudut “serang” kita dalam menyelesaikan suatu masalah pemrograman)..

1. **Paradigma prosedural** memandang penyelesaian masalah sebagai hasil dari serangkaian langkah yang menyelesaikan sub masalah
2. **Paradigma fungsional** memandang penyelesaian masalah sebagai komposisi dan aplikasi fungsi yang memetakan masalah ke jawaban
3. **Paradigma deklaratif** memandang penyelesaian masalah sebagai hasil inferensi terhadap fakta dan aturan yang diberikan
4. **Paradigma objek** memandang penyelesaian masalah sebagai hasil interaksi dari kelas yang membentuk objek (objek dalam konsep ini merupakan representasi objek di dunia nyata)

Struktur Dasar Algoritma

Struktur Dasar Algoritma

- Pernyataan (*statement*) → setiap langkah di dalam algoritma. Dikenal jg dengan istilah instruksi. Contoh:

  Tulis “Hello, world”

  Kalikan a dengan 2

- Langkah penyelesaian suatu masalah dapat berupa
  a. Runtunan aksi (*sequence*)
  b. Pemilihan aksi (*selection*)
  c. Pengulangan aksi (*repetition*)

Runtunan Aksi

- Terdiri dari 1 atau lebih pernyataan, setiap pernyataan ditulis dalam 1 baris atau dipisahkan dengan tanda titik koma.
- Tiap pernyataan dikerjakan secara berurutan (sekuensial) sesuai dengan urutannya di dalam teks algoritma, yaitu sebuah instruksi dilaksanakan setelah instruksi sebelumnya selesai dilaksanakan.
- Urutan instruksi menentukan keadaan akhir algoritma, bila urutan diubah, hasil akhir mungkin jg berubah.

Runtunan Aksi

- Contoh : pertukaran nilai dari dua peubah(variabel). Misal variabel A berisi nilai 8 dan variabel B berisi nilai 5. Jika ingin mempertukarkan nilai A dan B sehingga A bernilai 5 dan B bernilai 8, maka yang dilakukan :

  - Masukkan nilai A ke dalam C
  - Masukkan nilai B ke dalam A
  - Masukkan nilai C ke dalam B

Pemilihan

Adakalanya instruksi dikerjakan jika kondisi tertentu dipenuhi. Misal jika berada di perempatan yang ada traffic light. Jika lampu berwarna merah, maka kendaraan harus berhenti. Langkah ini dapat ditulis dalam pernyataan:

Jika lampu traffic light berwarna merah, maka berhenti

Sering disebut juga pernyataan kondisional

{jika ditulis **if**, dan **maka** ditulis **then}**

Contoh: if air di dalam ketel mendidih then

matikan api kompor

if x habis dibagi 2 then

tulis bahwa x bilangan genap

Pemilihan

- Jika ada pilihan aksi lainnya, maka dituliskan :

```
if kondisi then
    aksi 1
else
    aksi 2
```

Contoh:

```
if lampu A nyala then
    tekan tombol merah
else
    tekan tombol biru
```

- Perhatikan penggunaan **identasi** membantu kita memahami algoritma.

Pengulangan

- Bagaimana melakukan instruksi yang sama berulang kali?

  ALGORITMA tulis kalimat 500 kali:

  1. Tulis “saya anak yang rajin”
  2. Tulis “saya anak yang rajin”

  …..

  500. Tulis “saya anak yang rajin”
- Algoritma tidak elegan!

Pengulangan

ALGORITMA tulis kalimat 500 kali

    for i dari 1 sampai 500

        Tulis “saya anak yang rajin”

Struktur Teks Agoritma

1. Bagian Judul ➔ terdiri atas nama program dan penjelasan (spesifikasi) tentang program tersebut.
2. Bagian Deklarasi ➔ untuk mengumumkan semua nama yang dipakai di dalam algoritma beserta propertinya (misal tipe data).
3. Bagian Algoritma ➔ inti dari sebuah program, berisi instruksi-instruksi pemecahan masalah dalam notasi pseudo-code.

Contoh

```
PROGRAM HelloWorld
{ Program untuk mencetak “Hello world”. Masukan
program ini tidak ada. Keluarannya adalah tulisan
“Hello, world” tercetak di layar.}

DEKLARASI
     {tidak ada}

ALGORITMA
     print(“Hello, world”)
```

Contoh 2

```
PROGRAM FahrenheitCelcius

{ Program untuk mencetak tabel Fahrenheit-Celcius dari x sampai
y dengan kenaikan sebesar step. Masukan program ini adalah suhu
awal, suhu akhir, step, dan keluarannya adalah tabel konversi
suhu dalam C dan F}


DEKLARASI
      F, C : real
      x, y, step : integer
ALGORITMA
      read(x, y, step)
      F ← x
      while F ≤ y
            C = 5/9 * (F-32)
            print(F, C)
            F ← F + step
```

Referensi

- Munir, Rinaldi, *Algoritma & Pemrograman dalam Bahasa Pascal dan C* (Edisi Revisi), Informatika Bandung, 2011.
  - Bab 1 dan Bab 2