

# Pemrograman Dasar

Retno Mumpuni S.Kom, M.Sc

# Penilaian

- Keaktifan ( 10 %)
- Tugas ( 20 %)
- UTS ( 30%)
- UAS ( 40 %)

# Referensi

- Jeri R. Hanly, Elliot B. Koffman, Problem Solving and Program Design in C, 7th edition, Addison Wesley, 2012.
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Introduction to Algorithms, McGraw-Hill, 2003.

# Tools

- Menggunakan bahasa C
- IDE : Bloodshed Dev C++ ([www.bloodshed.net](http://www.bloodshed.net))
- Integrated Development Environment
- Bloodshed Dev C++ menggunakan MinGW sebagai compiler

# Program?

- Program adalah **kumpulan instruksi** yang digunakan untuk mengatur komputer agar melakukan suatu tindakan tertentu.
- Tanpa program yang dibuat pengguna, komputer tidak dapat melakukan apa-apa
- Konsep komponen komputer : hardware, software, **brainware**.
- Programmer : orang yang membuat program
- Programming / Coding : aktivitas yang berhubungan dengan pembuatan program

# Program = Kumpulan Instruksi

- Program ditulis dengan menggunakan kaidah / aturan bahasa pemrograman tertentu
- Sama halnya komunikasi antar manusia, komputer bisa menjalankan pekerjaannya sesuai dengan instruksi yang mengikuti kaidah/aturan.
- Secara garis besar, bahasa pemrograman dibagi :
  - High-level language : Bahasa beraras-tinggi
  - Low-level language : Bahasa beraras-rendah

# Low-level language

- Bahasa beraras rendah : bahasa pemrograman yang berorientasi pada mesin.
  - Menggunakan kode biner (0 atau 1), atau sebuah kode sangat sederhana untuk menggantikan kode tertentu dalam sistem biner.
  - Sangat sulit dipahami oleh orang awam, bahkan programmer sekalipun.
  - Sangat bergantung pada mesin (machine dependent)
  - Sangat cepat dieksekusi – komputer tidak perlu menerjemahkan terlalu jauh
  - MISAL :
    - B402 atau 1011 0100 0000 0100 artinya : inputkan angka 2 ke register AH
    - B22A atau 1011 0010 0010 1010 artinya : muatlah angka A2 hex ke register DL
    - CD21 atau 1100 1101 0010 0001 artinya : jalankan interupsi 21 heksadesimal
- 3 perintah diatas harus dieksekusi berurutan untuk sekedar menampilkan karakter \* pada layar**

# High-level language

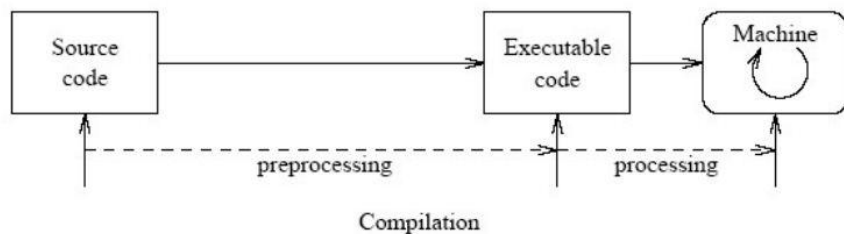
- Bahasa beraras tinggi adalah bahasa pemrograman yang berorientasi pada bahasa manusia.
- Dibuat dengan menggunakan bahasa yang mudah dipahami oleh manusia
- Contoh : Java, C, C++, Pascal, Basic, PHP, dll
- Misal :
  - Write (\*) -- PASCAL
  - Display "\*" – COBOL
  - PRINT "\*" – BASIC
  - Printf (\*) -- C
  - System.out.println(\*) – JAVA
  - Echo (\*) – PHP

Semua baris di atas adalah contoh instruksi untuk menampilkan karakter \* pada layar, dengan masing-masing contoh bahasa.

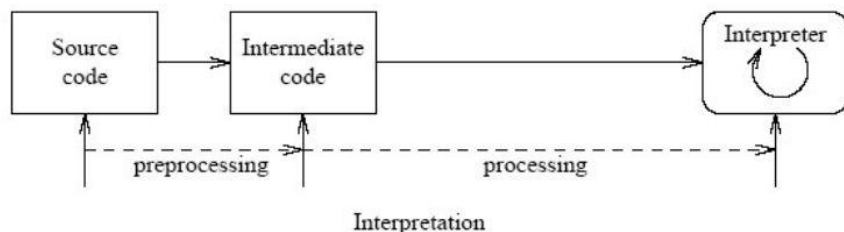


# Translator

- Sebelum bisa dieksekusi oleh mesin (komputer), high level language / bahasa beraras tinggi harus diubah / diterjemahkan terlebih dahulu ke bahasa mesin.
- Proses penerjemahan ini memerlukan **translator**.
- Berdasarkan urutan kerjanya, terdapat dua macam translator :
  - Compiler : C, C++, Pascal
  - Interpreter : Python, Matlab, JAVA



Di compiler, semua instruksi dalam high level **harus diterjemahkan secara utuh** terlebih dahulu dalam suatu executable code, untuk kemudian dieksekusi mesin. – misal : .exe file



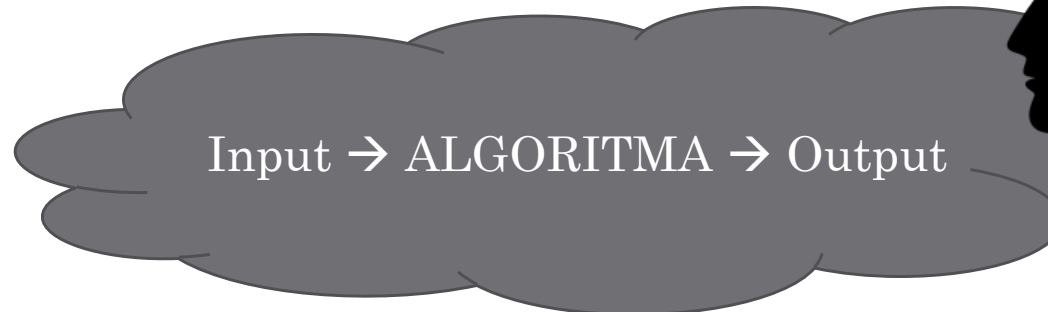
Di interpreter, **tiap baris instruksi dalam high level diterjemahkan** menjadi intermediate code per baris, untuk kemudian tiap baris intermediate code tersebut dieksekusi oleh mesin. – konsekuensinya : *lebih lambat*

# Interpreter vs Compiler

Interpreter	Compiler
<p>Kelebihan :</p> <p>Mudah untuk mencari kesalahan seandainya terdapat kesalahan pada program (debugging). Karena program dapat terus berjalan, hingga akhirnya komputer menemukan kesalahan.</p>	<p>Kelebihan :</p> <ol style="list-style-type: none"><li>1. Pengerjaan instruksi dilakukan sangat cepat, karena telah ditranslate ke bahasa mesin secara utuh.</li><li>2. Kode sumber tidak perlu didistribusikan ke pengguna yang menjalankannya, sehingga kerahasiaan terjamin.</li></ol>
<p>Kekurangan :</p> <ol style="list-style-type: none"><li>1. Kode sumber harus selalu tersedia – isu pencurian hak cipta</li><li>2. Eksekusi berjalan lambat</li></ol>	<p>Kekurangan :</p> <p>Kode sumber program yang ditulis harus benar seluruhnya secara sintaks, baru kemudian program dapat berjalan.</p>

# Membuat Program

- Program dibuat karena adanya masalah atau tugas yang akan diselesaikan
- Tahapan pembuatan program :
  - Menganalisis masalah
  - Membuat algoritma
  - Mengimplementasikan algoritma ke dalam instruksi program
  - Mengeksekusi dan menguji program
- Menganalisis masalah adalah tahapan yang penting. Diperlukan pengalaman, pengetahuan, imajinasi, kreativitas, dan kecerdasan untuk menganalisa informasi apa yang akan menjadi input dan output, serta bagaimana mengolahnya.



analisa



# Algoritma

- Untuk mengolah data menjadi informasi (input menjadi output), programmer wajib menyusun langkah detail (runutan) bagaimana komputer akan menyelesaikan masalah tersebut ⇔ Langkah detail ini disebut **Algoritma**
- Istilah algoritma berasal dari seorang ilmuwan Persia : Al-Khwarizmi (790 – 840 ) – bapak aljabar.



# Contoh Algoritma







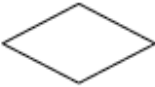


- Menghitung Luas Lingkaran
  - $3.14 \times \text{jari-jari}^2$
- Algoritma / runutan langkahnya adalah sebagai berikut :
  - Dapatkan jari-jari lingkaran
  - Hitung luas lingkaran dengan menggunakan rumus  $3.14 \times \text{jari-jari}^2$
  - Tampilkan nilai luas lingkarannya
- Selain dengan kalimat, algoritma juga bisa dinyatakan dalam **pseudocode**
- Misal :
  - Hitung luas lingkaran dengan menggunakan rumus  $3.14 \times \text{jari-jari}^2$   
dapat dinyatakan dengan pseudocode berikut  
 $\text{Luas} \leftarrow 3.14 * (\text{jari-jari})^2$

# Pseudocode

- Menghitung luas lingkaran
1. Jari-jari  $\leftarrow 20$
  2. Luas  $\leftarrow 3.14 * (\text{Jari-jari}^2)$
  3. Print (Luas)

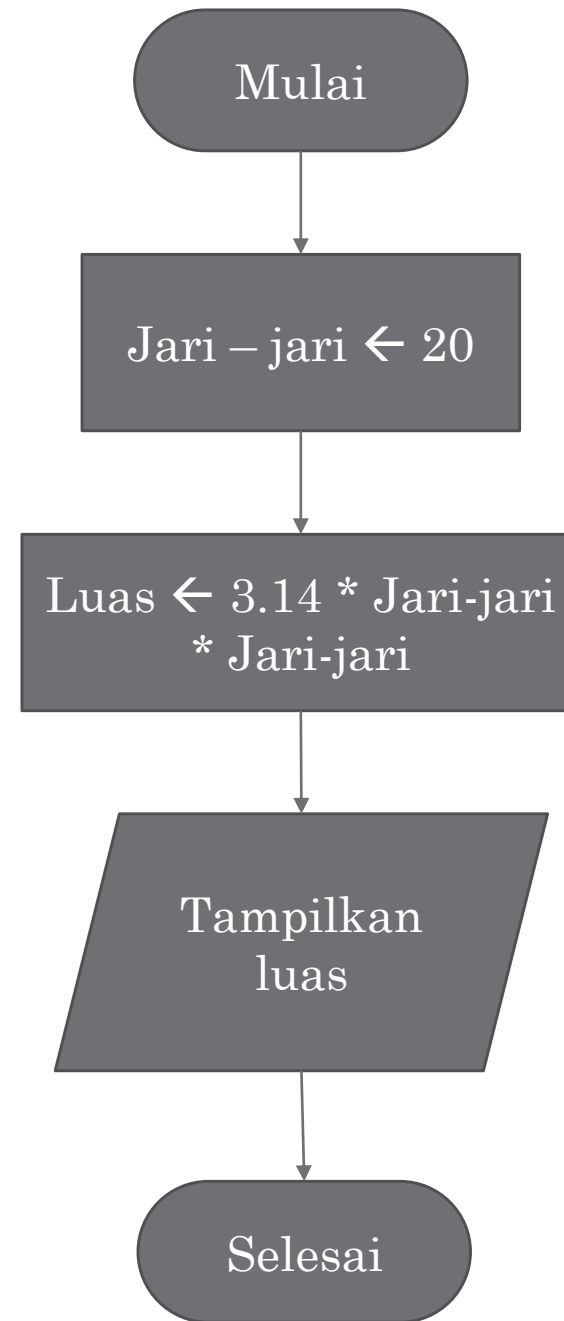
# Flowchart

- Selain dinyatakan dalam pseudocode, algoritma juga bisa direpresentasikan dalam flowchart (diagram alir).

SIMBOL	NAMA	FUNGSI
	<b>TERMINATOR</b>	Permulaan/akhir program
	<b>GARIS ALIR (FLOW LINE)</b>	Arah aliran program
	<b>PREPARATION</b>	Proses inisialisasi/ pemberian harga awal
	<b>PROSES</b>	Proses perhitungan/ proses pengolahan data
	<b>INPUT/OUTPUT DATA</b>	Proses input/output data, parameter, informasi
	<b>PREDEFINED PROCESS (SUB PROGRAM)</b>	Permulaan sub program/ proses menjalankan sub program
	<b>DECISION</b>	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
	<b>ON PAGE CONNECTOR</b>	Penghubung bagian-bagian flowchart yang berada pada satu halaman
	<b>OFF PAGE CONNECTOR</b>	Penghubung bagian-bagian flowchart yang berada pada halaman berbeda

# Contoh Flowchart

- Menghitung Luas Lingkaran





# Latihan Flowchart

- Buatlah flowchart menghitung keliling persegi panjang
- Buatlah flowchart dari vending machine
- Buatlah flowchart dari mesin ATM
  
- (maju ke papan tulis)

# Contoh Program

```
#include <stdio.h>
```

```
int main ( )
```

```
{
```

```
    double jari_jari;
```

```
    double luas;
```

```
    jari_jari = 20;
```

```
    luas = 3.14 * jari_jari * jari_jari;
```

```
    printf("Luas lingkaran = %lf", luas);
```

```
    return 0;
```

```
}
```