

Pengantar Dunia Pemrograman

Tim Pengajar IF1210

Sekolah Teknik Elektro dan Informatika

Tujuan

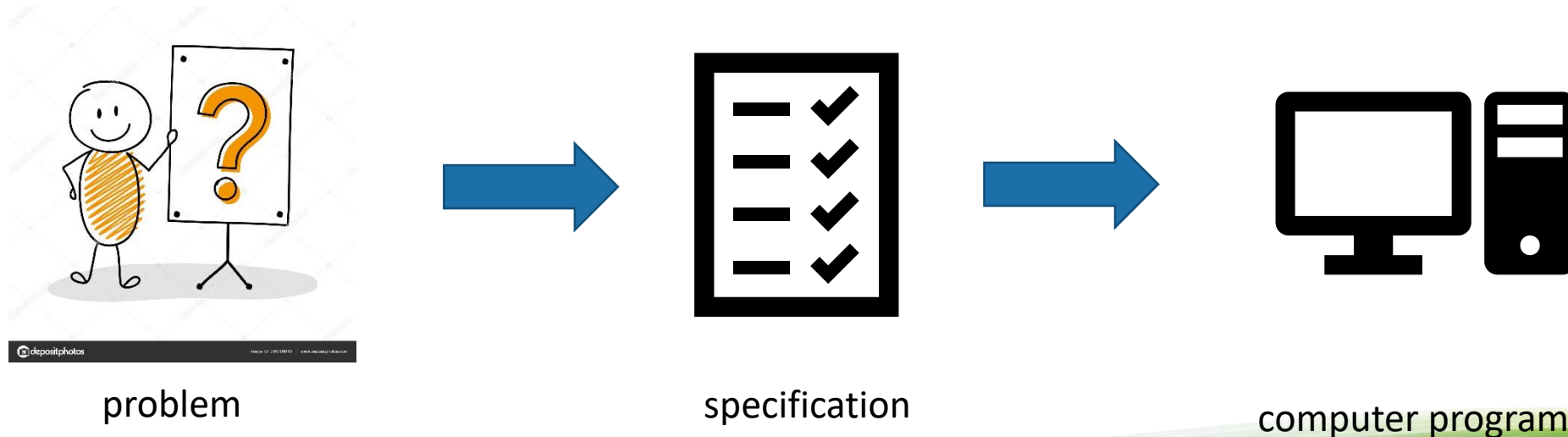
- Mahasiswa dapat memahami:
 - Apa itu pemrograman komputer
 - Paradigma pemrograman
 - Bahasa pemrograman
 - Berbagai hal terkait pemrograman (lingkungan, pemroses, dll.)
 - Pemrograman vs *software engineering*

Pemrograman

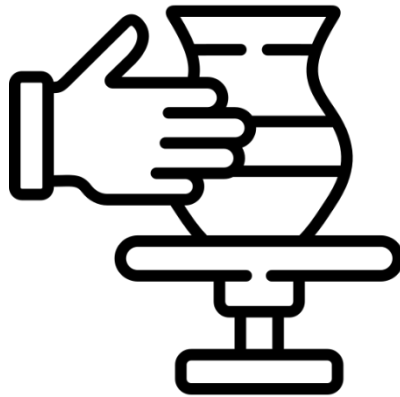
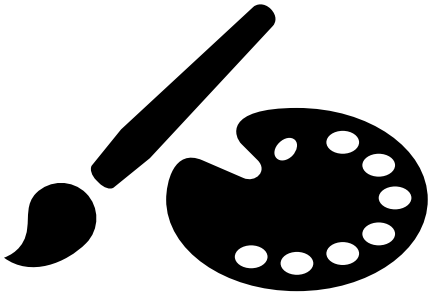
Pemrograman [komputer] adalah proses untuk memformulasi persoalan komputasi menjadi program [komputer]

Pemrograman tidak hanya *coding*

Pemrograman adalah analisis persoalan (membuat spesifikasi), implementasi (*coding*), *testing*, *debugging*



Programming: *art/craft/engineering?*



Kegiatan Pemrograman



Analisis persoalan, membuat spesifikasi, menyusun algoritma



Program writing (coding), yaitu implementasi pada bahasa pemrograman tertentu



Program execution (observation, debugging, testing)



Program reading



Program correctness and complexity analysis



Program maintenance

Belajar Pemrograman = Belajar Bahasa Pemrograman??

- Ada **RIBUAN** bahasa pemrograman di dunia saat ini!!
- Tidak mungkin semua bahasa pemrograman dipelajari di kuliah

Belajar **pemrograman** =
belajar **pola pikir komputasional** + **paradigma pemrograman**

Belajar memrogram \neq belajar bahasa pemrograman

Paradigma Pemrograman

Paradigma Pemrograman

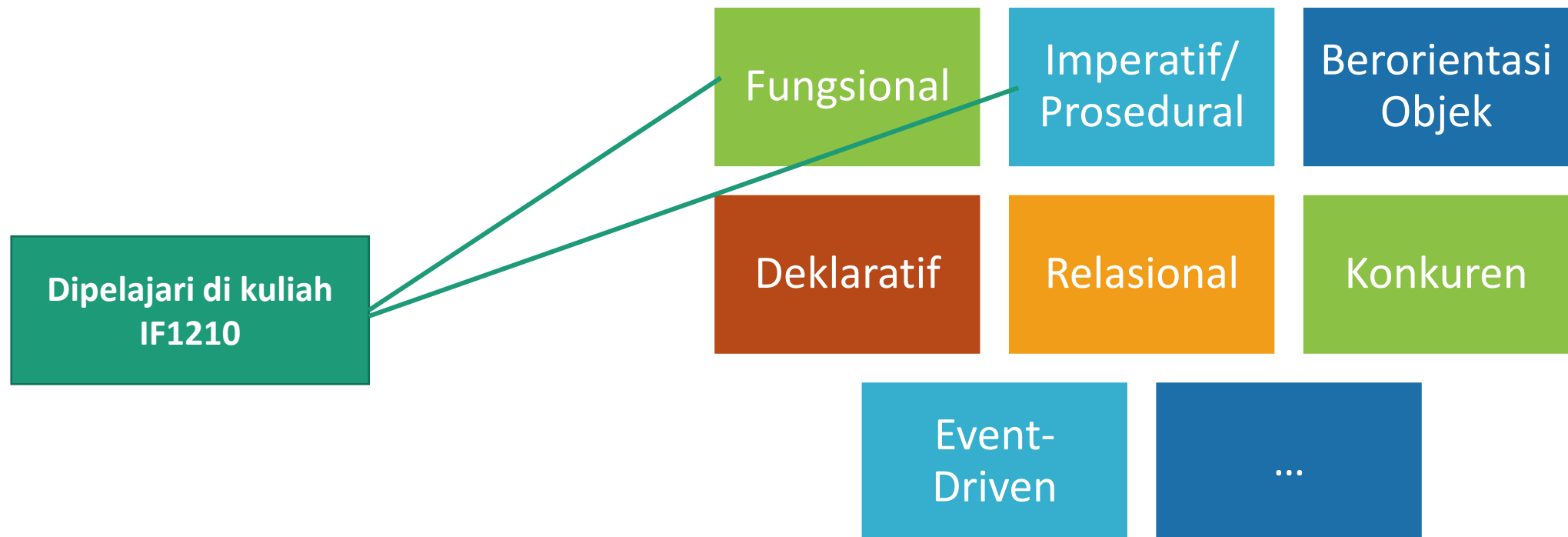
- **Paradigma** [pemrograman] adalah sudut pandang penyelesaian persoalan dengan [program]
- Setiap persoalan menggiring kita pada **pendekatan khusus** untuk pemecahannya
- Paradigma memberikan strategi analisis khusus pemecahan masalah
- Jenis persoalan tertentu dapat dipecahkan dengan baik dengan paradigma tertentu



What do you see?
By shifting perspective you might see an old woman or a young woman.

Paradigma Pemrograman

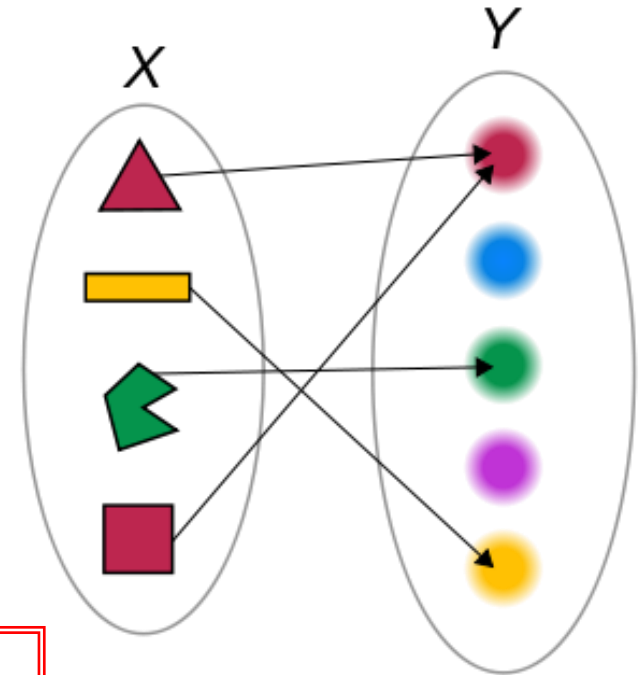
Jenis **paradigma pemrograman** yang dikenal:



Berikut ini diperkenalkan beberapa di antaranya...

Paradigma Fungsional

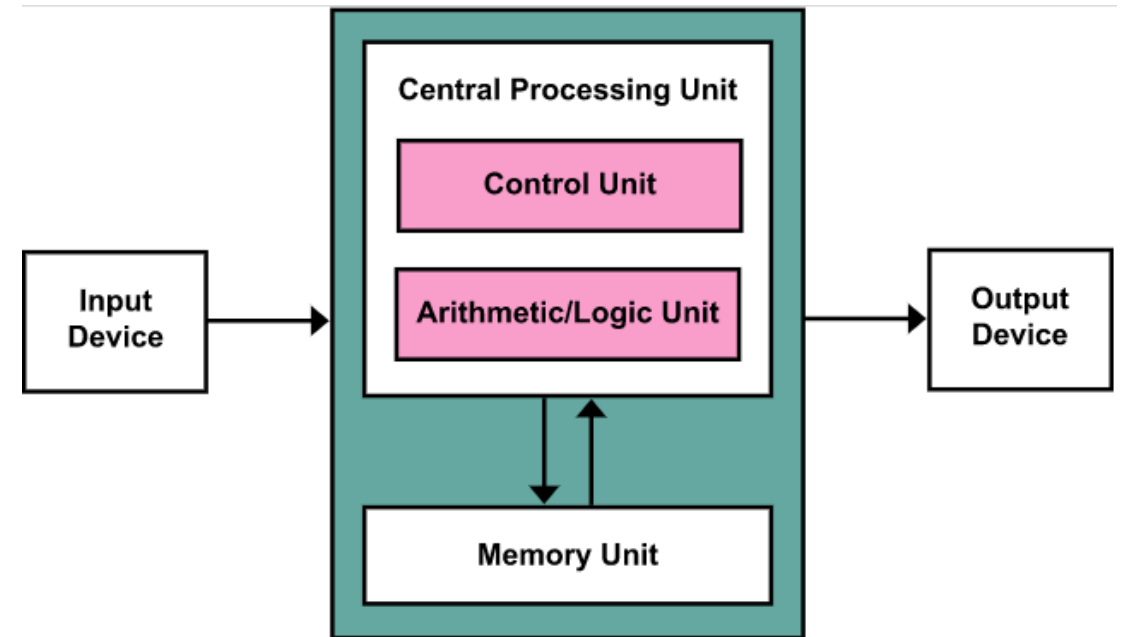
- Didasari oleh konsep pemetaan dan **fungsi** di matematika
- Pemrogram mengasumsikan bahwa ada **fungsi-fungsi** yang terdefinisi
- Penyelesaian masalah didasari atas **aplikasi dari fungsi-fungsi**



Paradigma Imperatif/Prosedural

- Didasari oleh konsep mesin Von Neumann (*stored program concept*)
- Program didasari oleh **strukturisasi informasi** di dalam memori dan **manipulasi** dari informasi yang disimpan tersebut

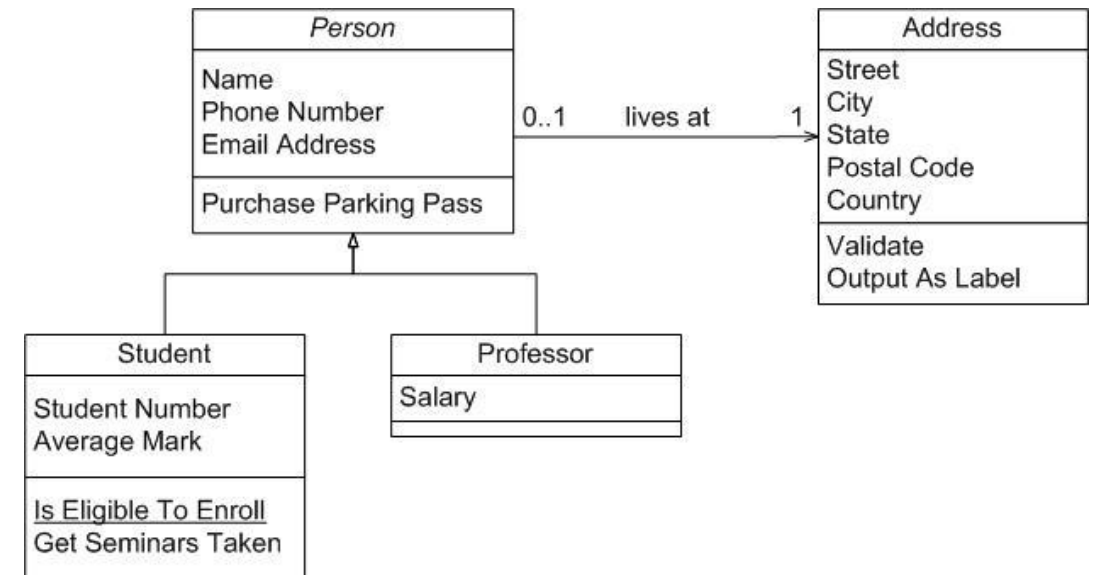
Program = Algoritma + Struktur Data



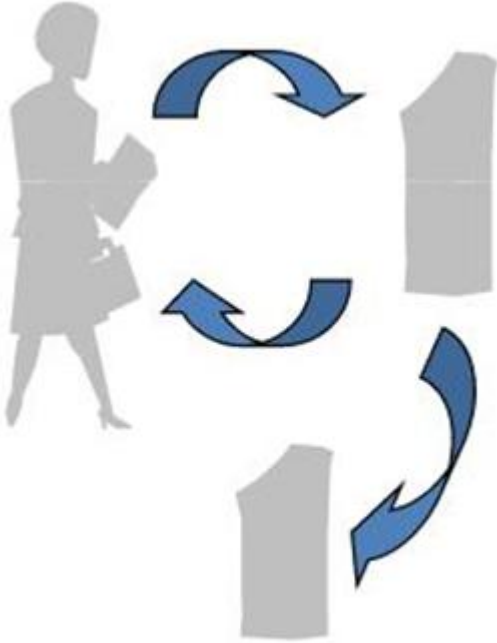
Skema arsitektur mesin Von Neumann

Paradigma Berorientasi Objek

- Didasari oleh **konsep objek**
- Sebuah objek mempunyai **atribut** (kumpulan sifat) dan mempunyai **kelakuan** (kumpulan reaksi, metoda)
- **Kelas** adalah cetak biru dari objek-objek dengan atribut dan kelakuan yang sama



Paradigma Prosedural



Withdraw, deposit, transfer

Paradigma Berorientasi Objek



Customer, money, account

Paradigma
Deklaratif

Paradigma
Relasional

Paradigma
konkuren

Paradigma
Event-Driven

Paradigma fungsional

- Haskell, LISP, Scheme, Erlang, Scala, Miranda, ...

Paradigma Prosedural

- Basic, C, Pascal, Ada, Fortran, COBOL, ...

Paradigma berorientasi objek

- Eiffel, SmallTalk, Java, C++, C#, ...

Paradigma relasional

- SQL

Paradigma deklaratif

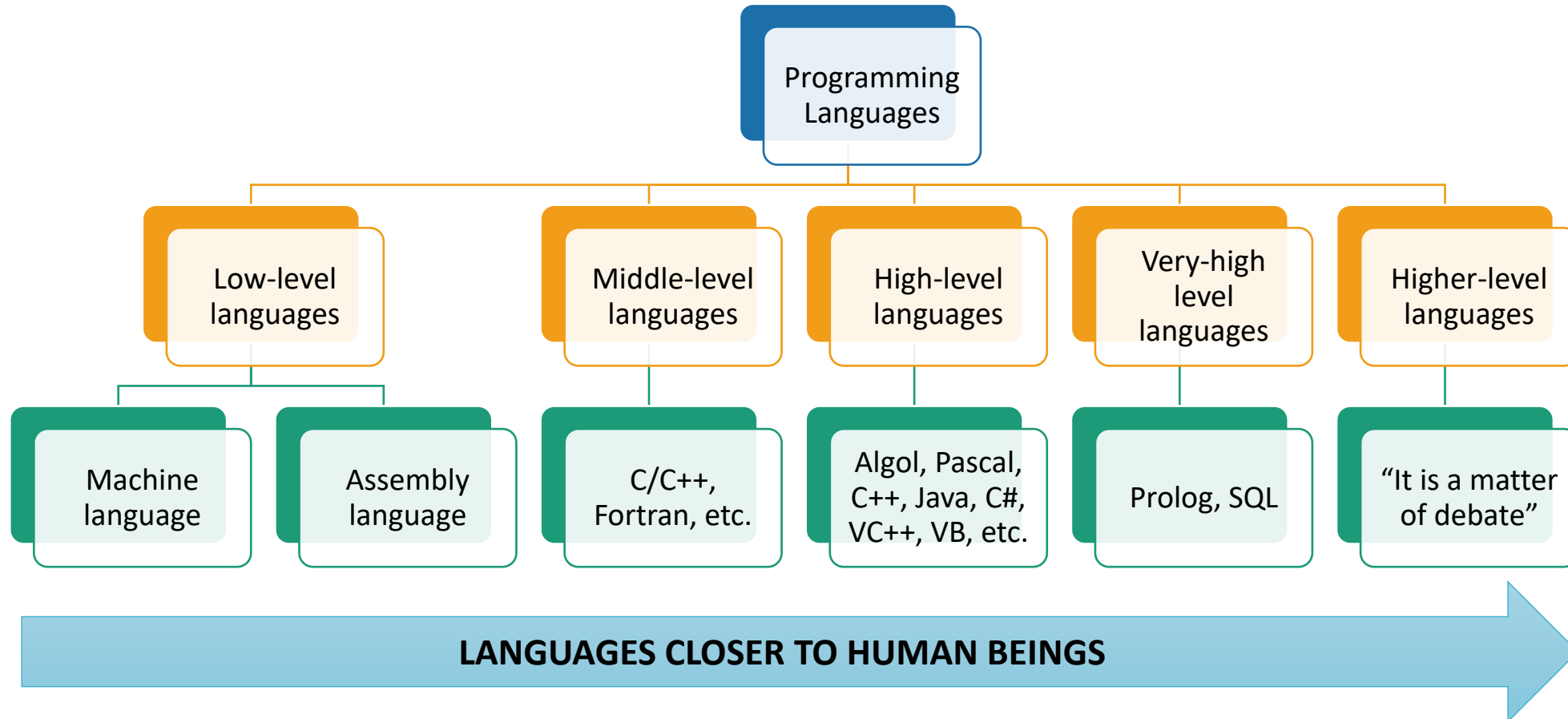
- Prolog

Multi-paradigm

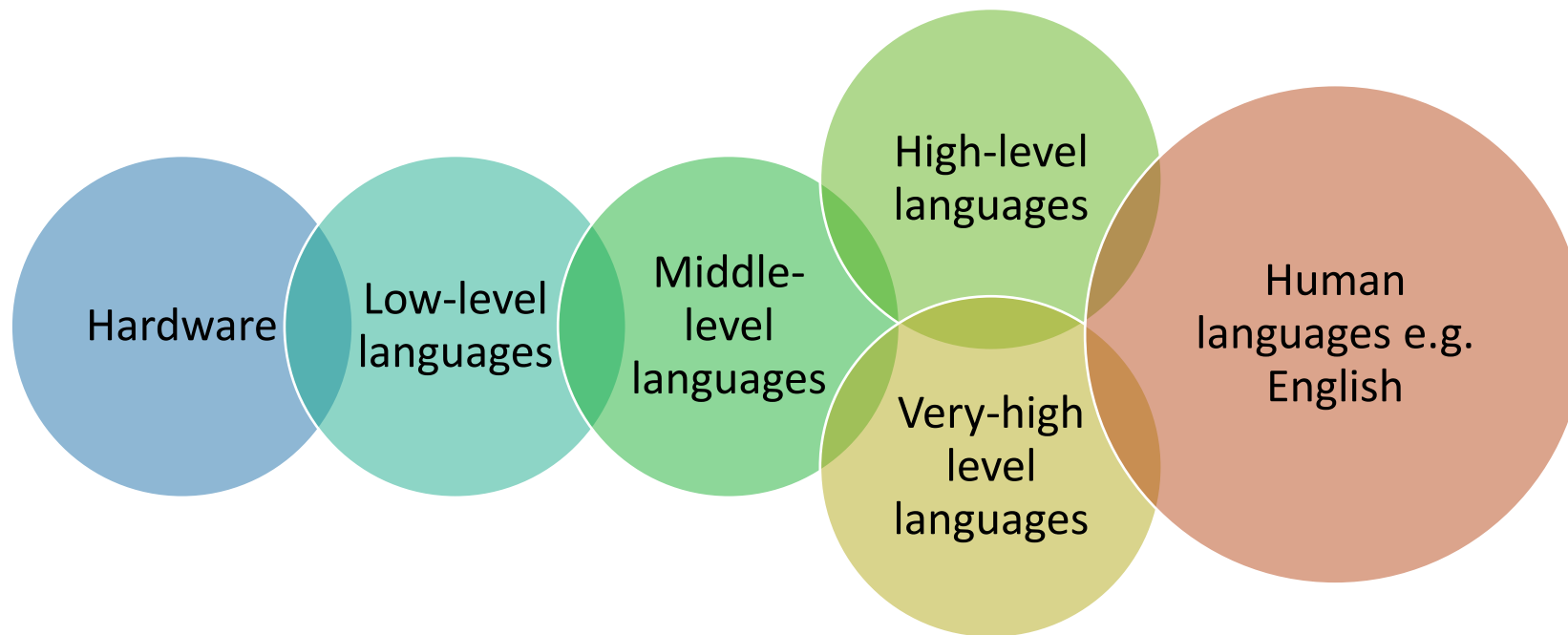
- Java, Python

Bahasa Pemrograman

Taksonomi Bahasa Pemrograman



Overlapping of Languages



Machine Language

MIPS32 Add Immediate Instruction

001000	00001	00010	0000000101011110
OP Code	Addr 1	Addr 2	Immediate value

Machine code

Equivalent mnemonic: **addi** \$r1, \$r2, 350

Efficient code
for the
machine

Fast
processing

Hardware/
machine
dependent

Tedious and
error prone

Difficult to use
or debug, to
understand

Assembly Language

<i>Language Code (Machine) (16-BIT INSTRUCTION SET)</i>	<i>Assembly Language Code (Equivalent)</i>			
1000000100100101	LOAD	R1	5	
1000000101000101	LOAD	R2	5	
1010000100000110	ADD	R0	R1	R2
1000001000000110	SAVE	R0	6	
1111111111111111	HALT			

Efficient code for the machine

Fast processing

Easier to understand, to debug, to modify compared to machine language

Need at translator for the execution of the program

Hardware/machine dependent

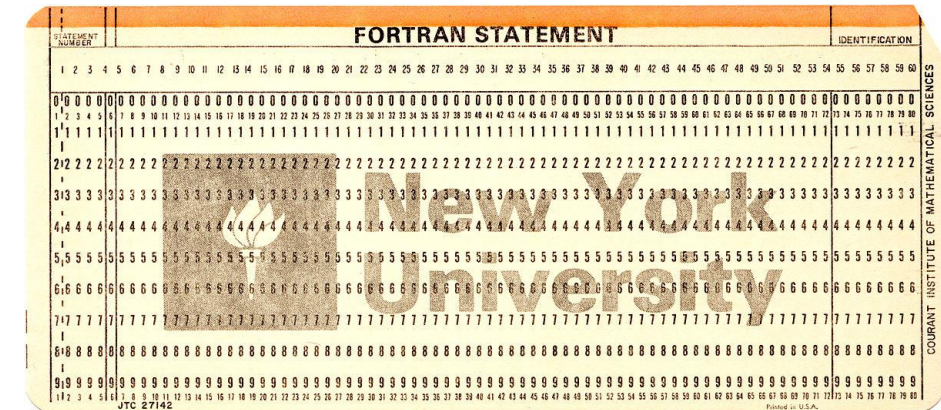
Used for specific applications

Middle-Level Languages

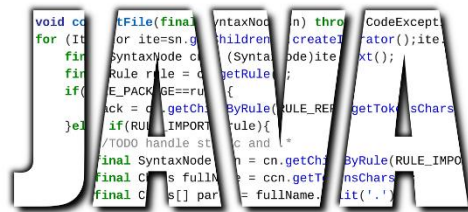
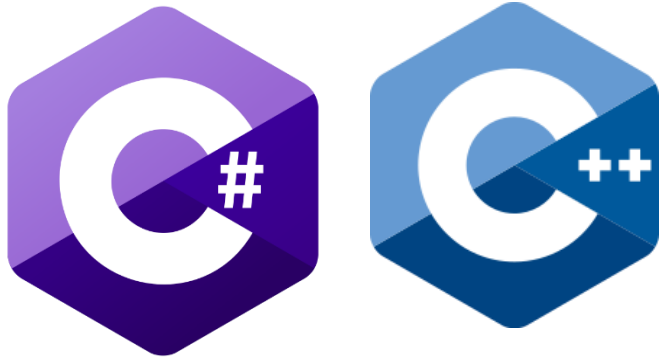
Bridge the gap of high-level and low-level languages

Need more technical skills compared to the high-level language

Providing a small set of controlling and data-manipulating instructions



High Level Languages



Problem-oriented languages

Easy to use, to understand, to debug

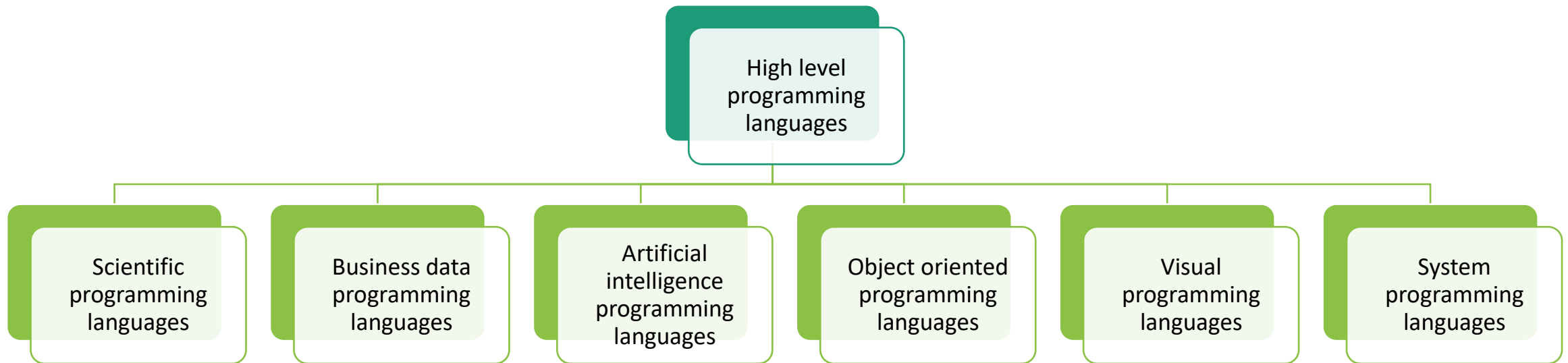
Portable, machine independent

User friendly, easy to write programs

Need a translator to machine level

Less efficient

High Level Programming Languages



Very High Level Languages

4GL, mostly non
procedural

Multiply the numbers A and B
And put the result into C

the users and the
developers to
describe the
results they need

Examples:



Wolfram Mathematica

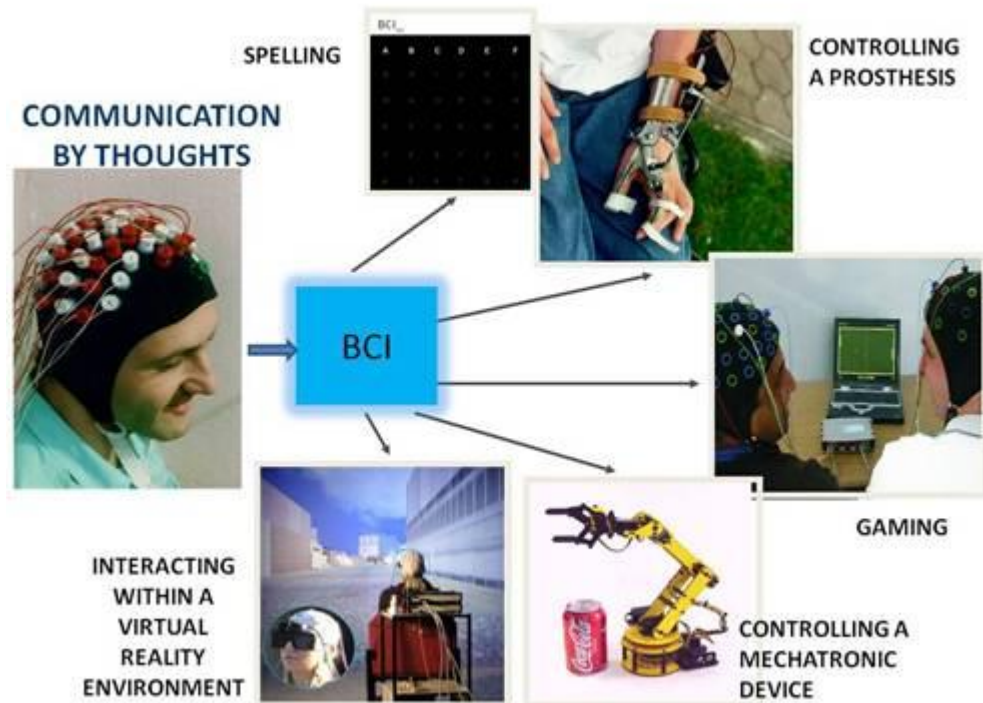


Oracle 4GE

Higher Level Prog. Languages???

5GL → yet to come

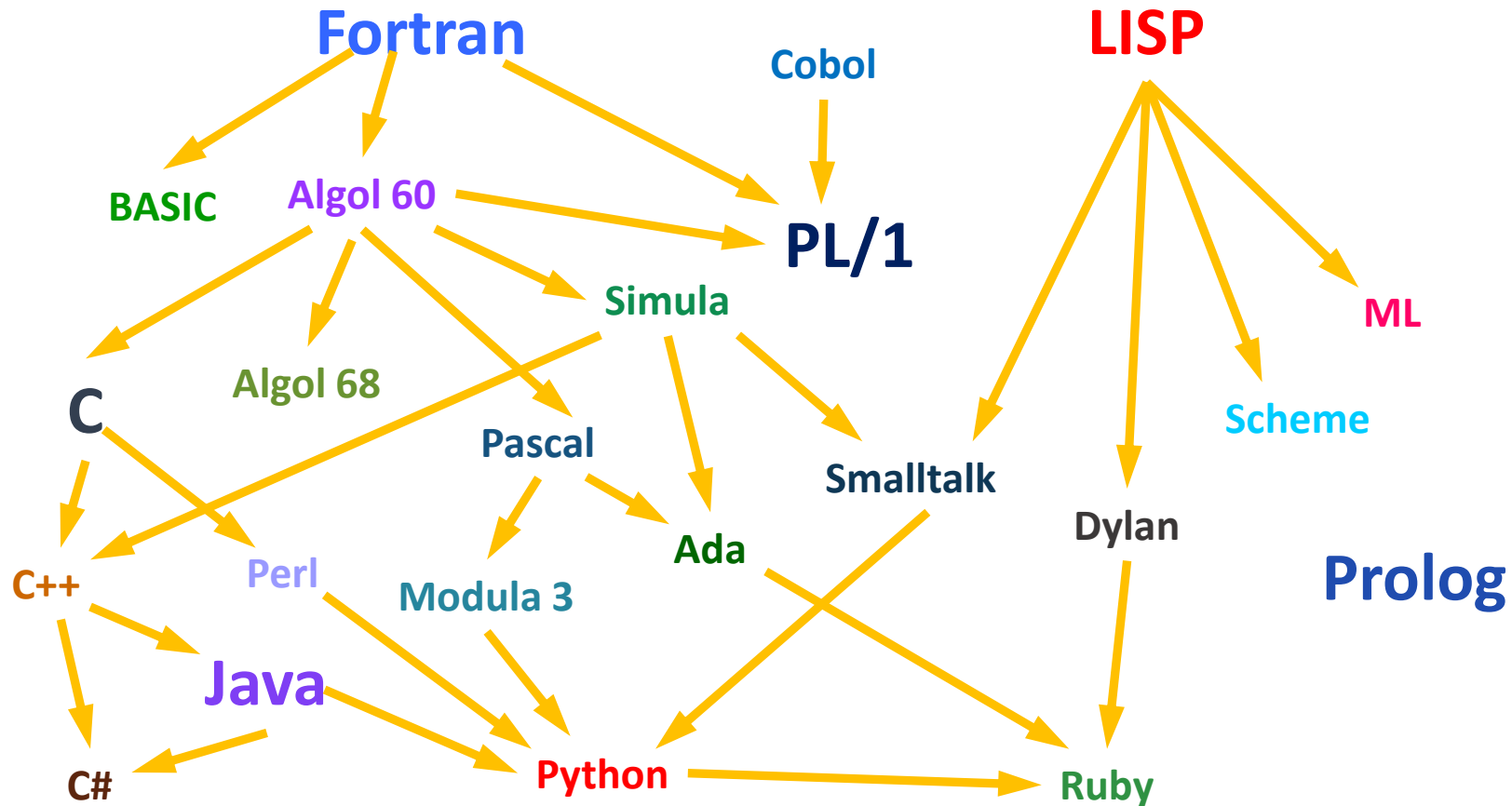
Interface between
human being and
machine to permit
effective use of natural
language and image



Brain Computer Interface

Intermezzo...

A family tree of languages



Intermezzo

- Bahasa pemrograman **Pascal** diberi nama berdasarkan matematikawan dan filosofer Perancis, Blaise Pascal
- Bahasa pemrograman **Ada** diberi nama berdasarkan Ada Lovelace
- Bahasa pemrograman **C** merupakan pengembangan dari bahasa pemrograman **B**
- Bahasa **Java** asalnya diberi nama *Oak* (dari pohon ek yang berdiri di depan kantor pembuatnya), lalu diubah menjadi *Green*, lalu baru menjadi *Java* (dari *Java coffee*, yang banyak diminum oleh para pembuatnya)
- Nama Bahasa **Python** adalah *tribute* ke grup pelawak Britania, Monty Python.
 - *Therefore: "An important goal of Python's developers is keeping it fun to use"*

Aspek-Aspek Lain Pemrograman

Artefak utama

```
40
41 $(function){cards();});
42 $(window).on('resize', function(){cards();});
43 function cards(){
44   var width = $(window).width();
45   if(width < 750){
46     cardssmallscreen();
47   }else{
48     cardsbigscreen();
49   }
50 }
51 function cardssmallscreen(){
52   var cards = $(''.card').length;
```

Source code [+dokumentasi]

Lingkungan Pemrograman: Editor Source Code

```

1  #include <stdio.h>
2
3  int main(void)
4  {
5      printf("hello, world\n");
6      return 0;
7  }

```

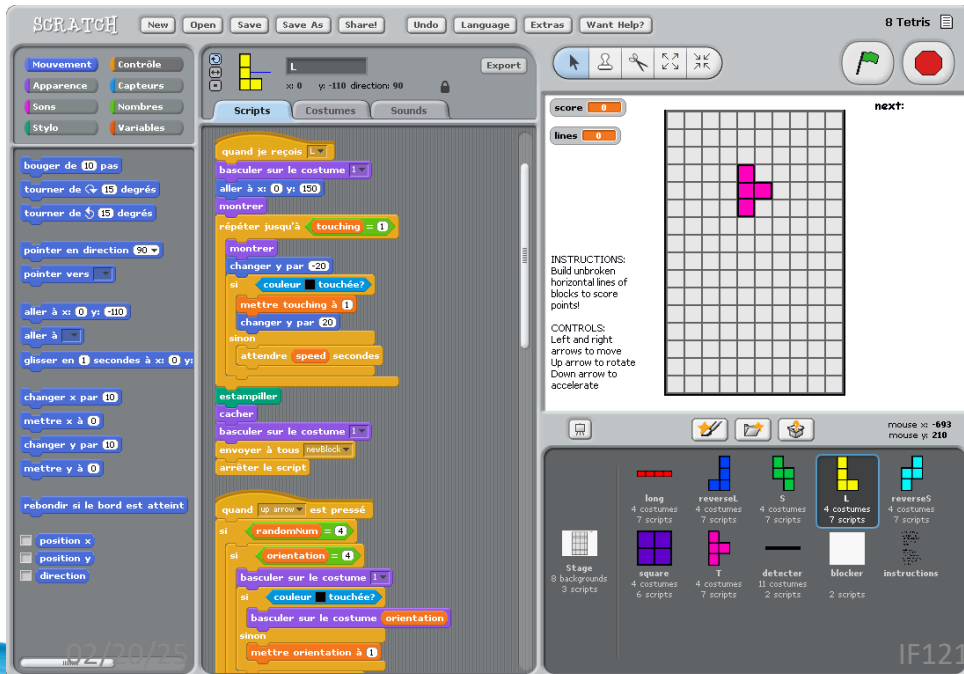
Text editor

```

26  int main(void)
27  {
28      unsigned char c;
29      int quit = 0;
30
31      InitLED();
32      puts(my_str);
33      puts("Your choice please\n");
34
35      while (!quit)
36      {
37          c = getchar();
38
39          switch (c)
40          {
41              case '0':
42                  LED_OFF;
43                  puts(" LED switched to OFF\n");
44                  break;
45              case '1':
46                  LED_ON;
47                  puts(" LED switched to ON\n");
48                  break;
49          }
50      }

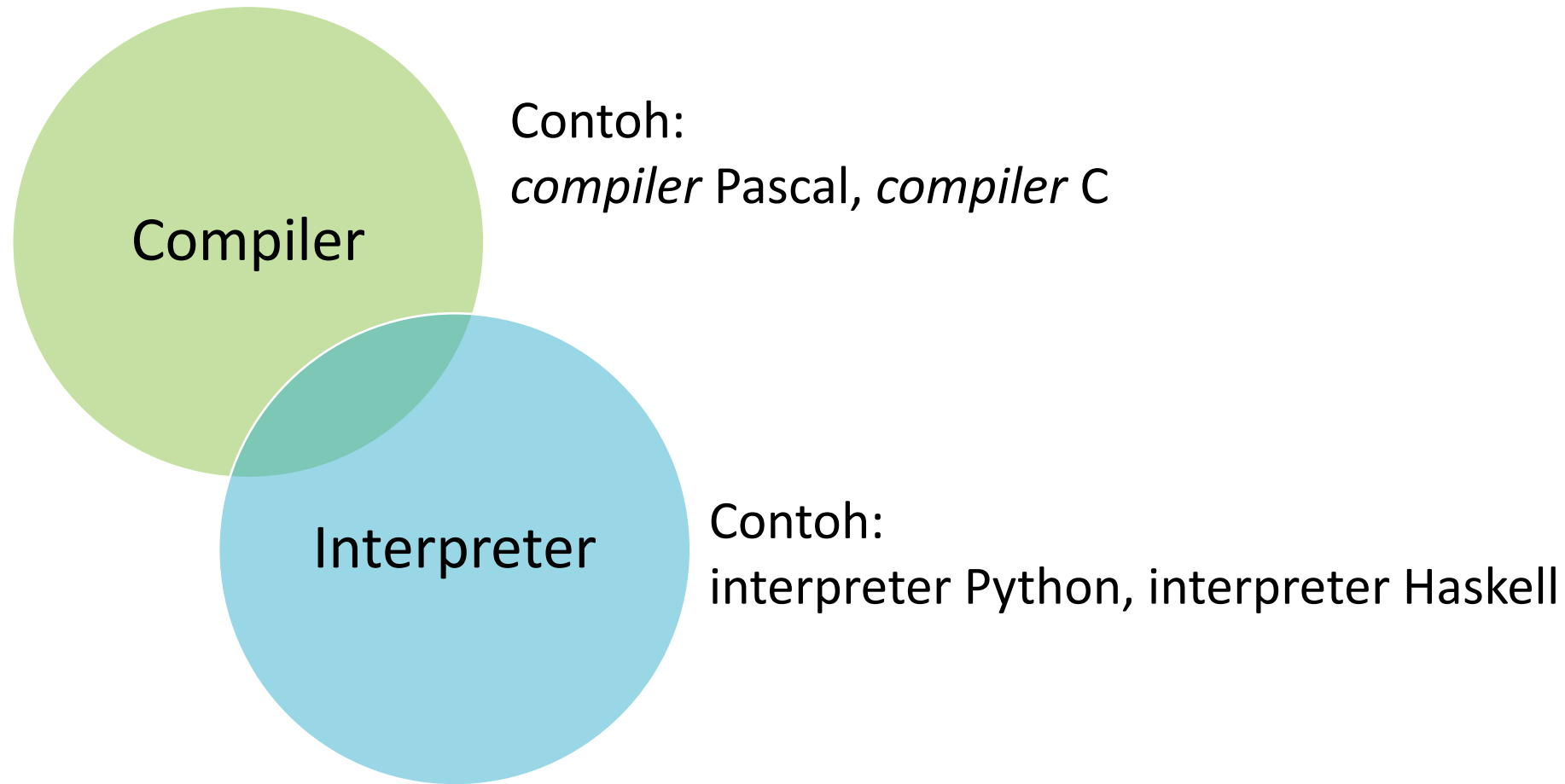
```

Contoh IDE: Code::Blocks

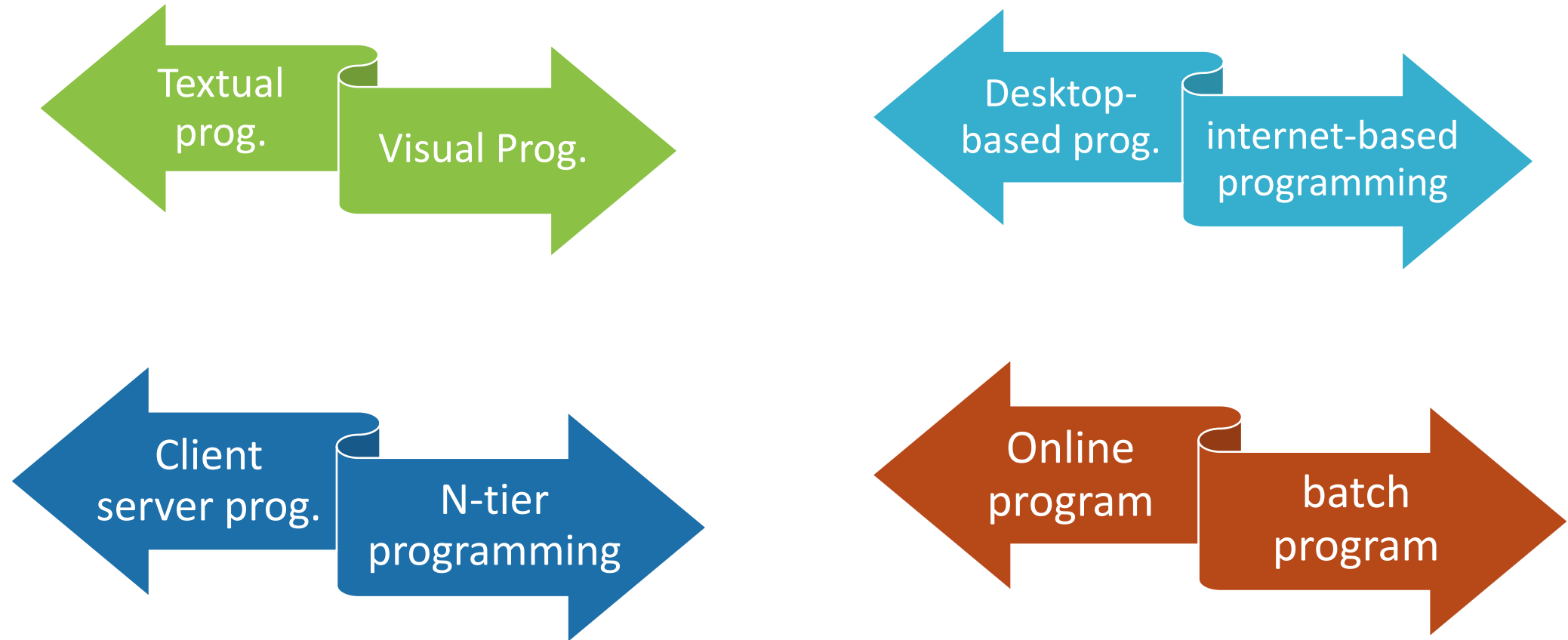


Visual programming

Lingkungan Pemrograman: Pemroses Bahasa



Berbagai Area Pemrograman



Skala program [relative]

Program skala kecil

Program sedang

Program besar

Kompleksitas program

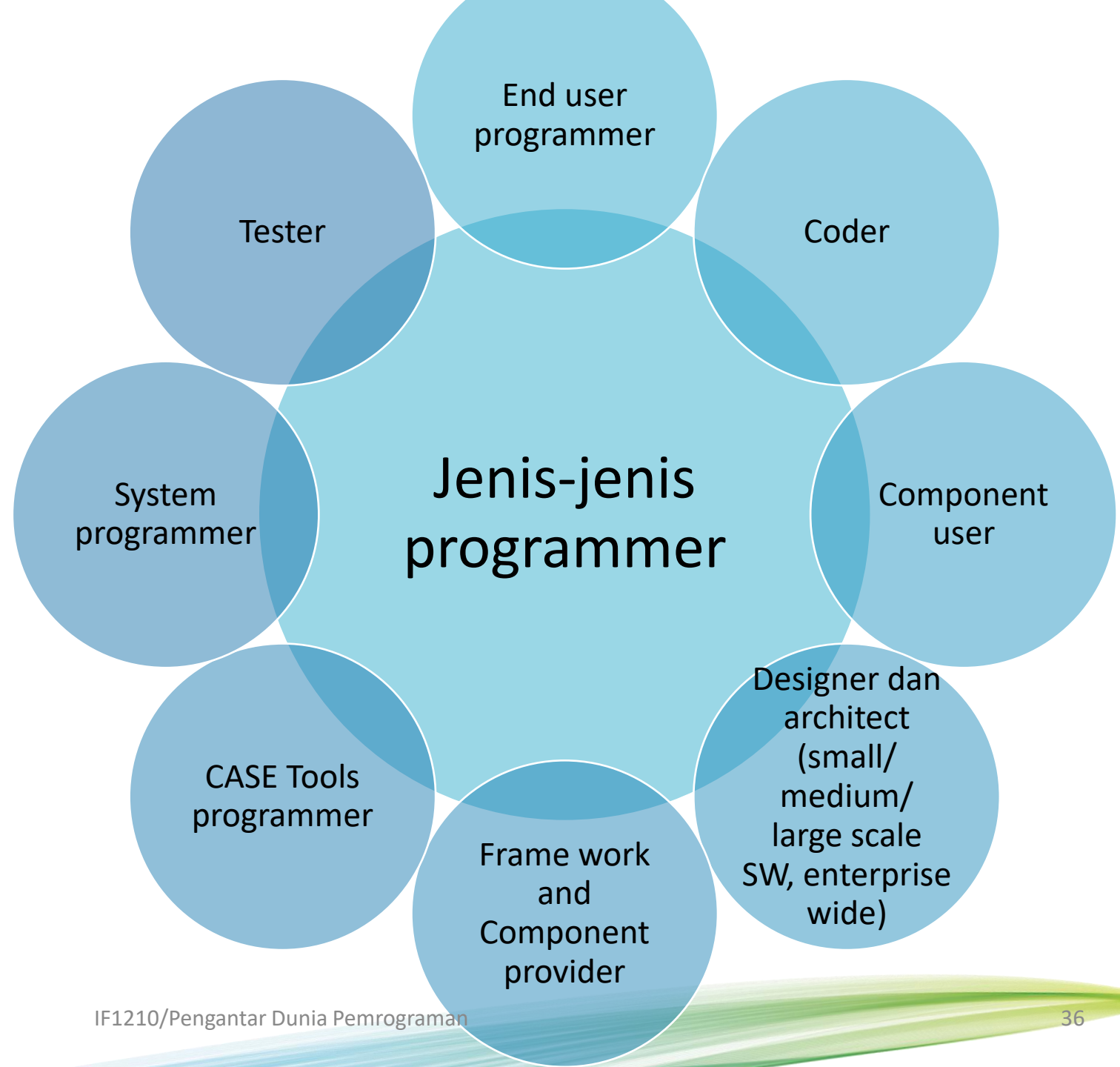
Algoritma dasar

Algoritma lanjut

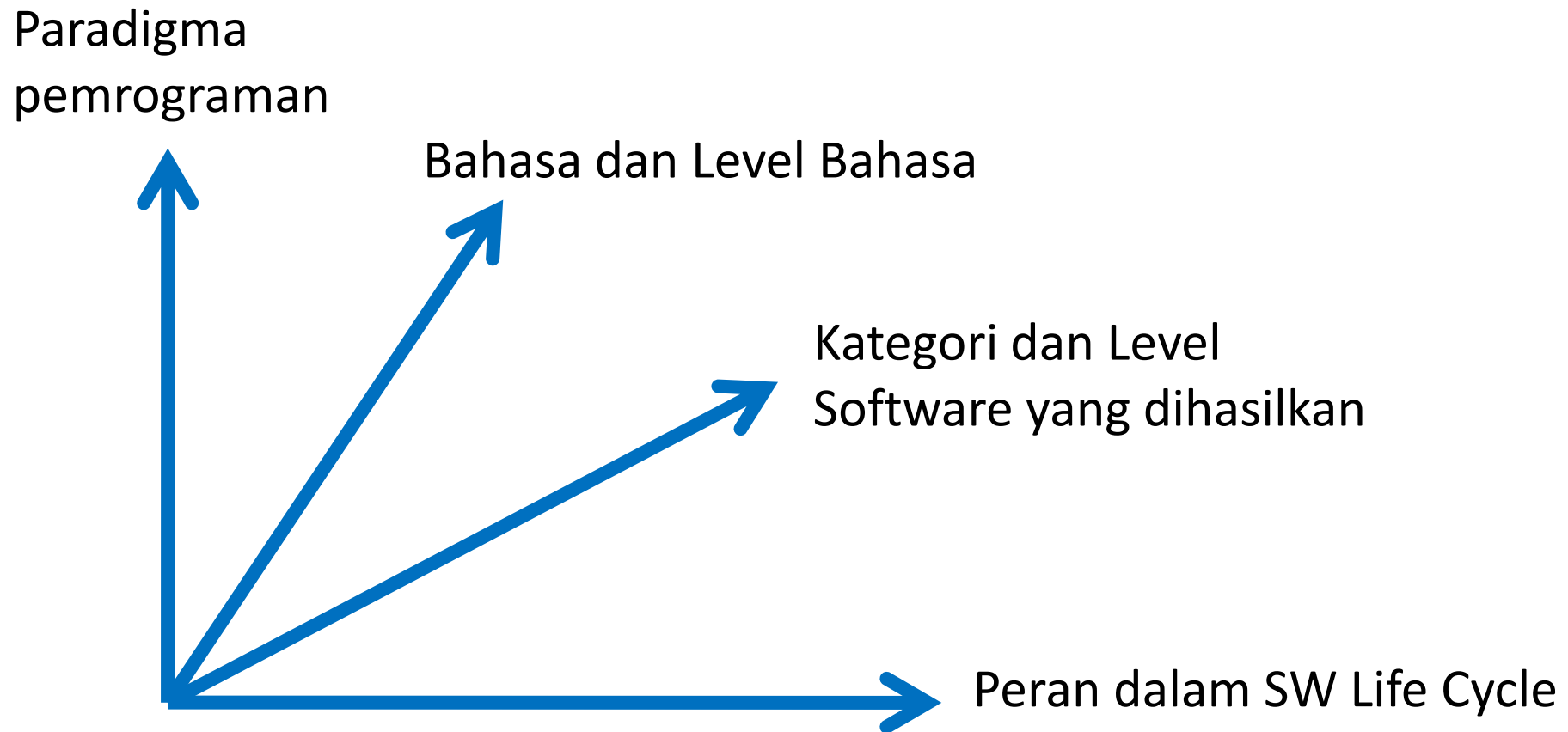
- Dynamic programming
- Branch and bound
- Advanced search
- Advanced data structure

Kategori Persoalan Pemrograman

- mudah dikerjakan (*tractable*) dengan solusi yang efisien dalam waktu yang wajar
- tidak mudah dikerjakan (*intractable*)
- dapat diselesaikan dengan pendekatan tetapi tidak optimal (*solvable approximately, but not optimally*)
- belum memiliki solusi yang efisien
- tidak dapat diselesaikan (*not solvable*)



Taksonomi Programmer

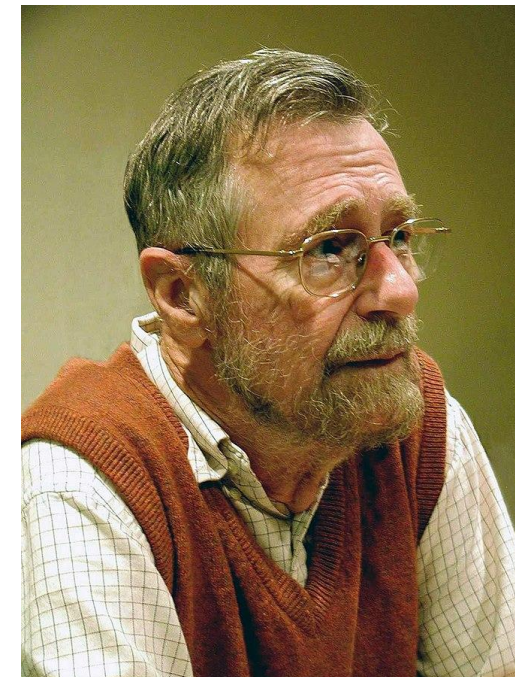


Level Programmer 😊

- Dead Programmer:
 - Dijkstra, Kay
- Successful Programmer:
 - Gates, Carmack, DHH
- Famous Programmer
- Working Programmer
- Average Programmer
- Amateur Programmer
- Unknown Programmer
- Bad Programmer



David Heinemeier Hansson



Edsger Dijkstra



John D. Carmack

<https://blog.codinghorror.com/the-eight-levels-of-programmers/>

Pemrograman dan Software Engineering

Definisi formal ***software engineering***:

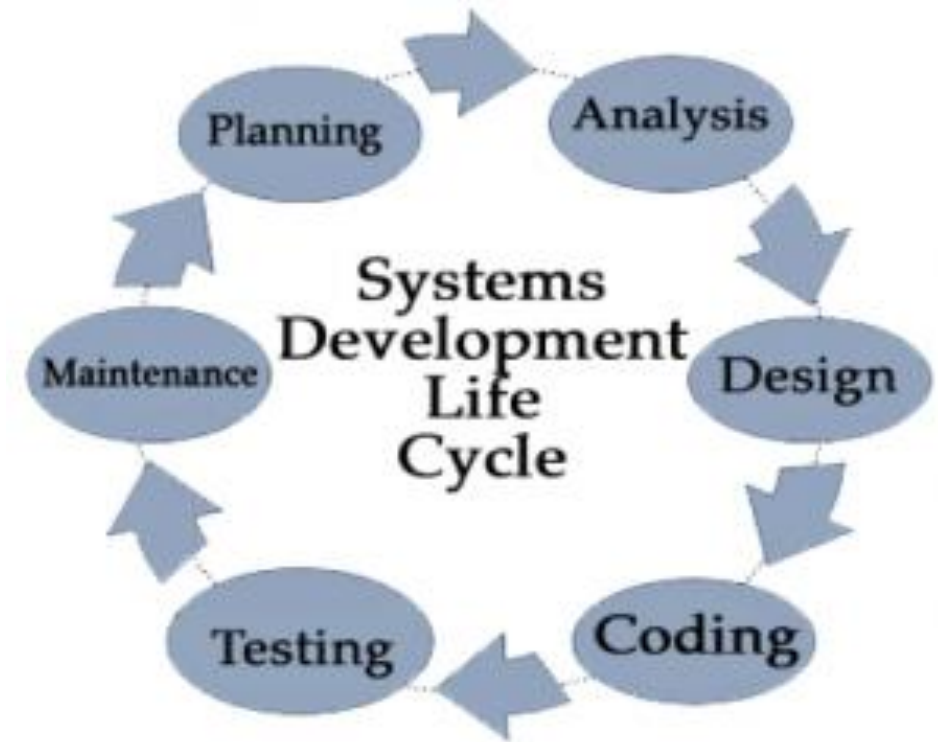
“the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software”

IEEE Standard Glossary of Software Engineering Terminology,” IEEE std 610.12-1990, 1990

Pemrograman adalah bagian dalam proses *software engineering* (rekayasa perangkat lunak)

Software Engineering Life Cycle

- Requirement analysis
- Software analysis and design
 - Kegiatan pemrograman: Analisis dan penentuan spesifikasi program
- Implementation (coding and debugging)
 - Kegiatan pemrograman: coding dan debugging
- Unit and component testing
 - Kegiatan pemrograman: testing
- Integration and System testing
- Maintenance





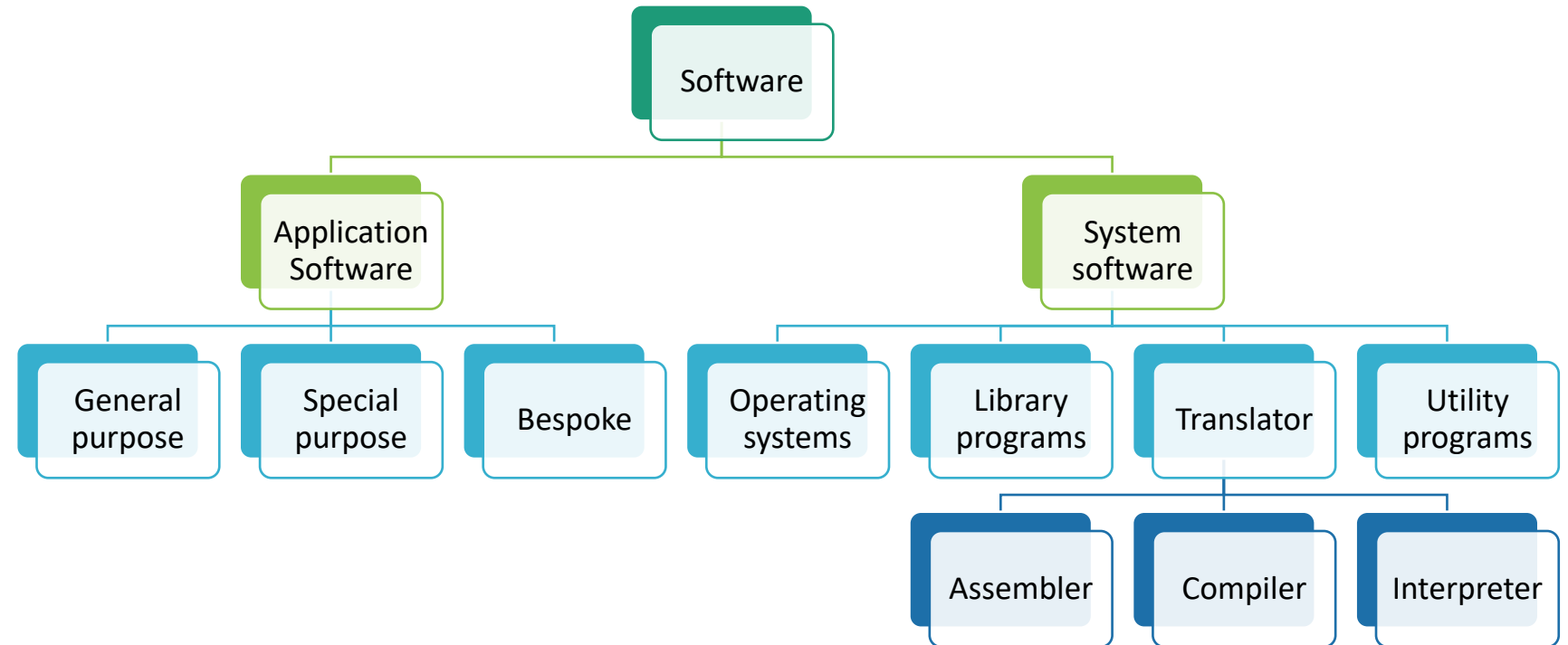
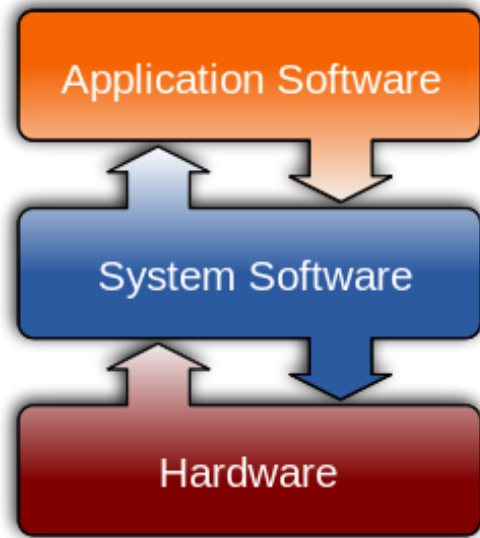
Software diibaratkan pencakar langit

VS



Program [kecil] diibaratkan rumah kecil

Kategori Software



SYSTEM SOFTWARE

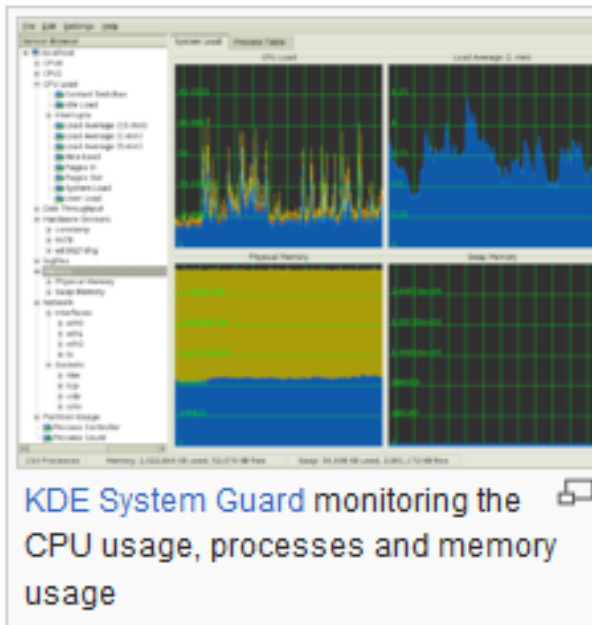
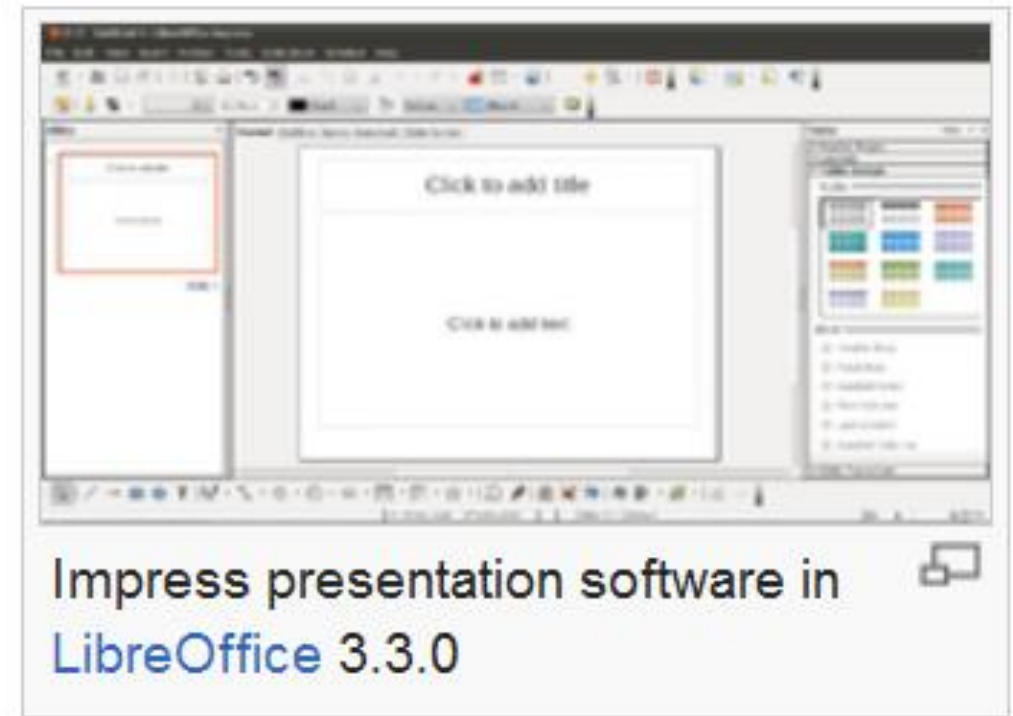
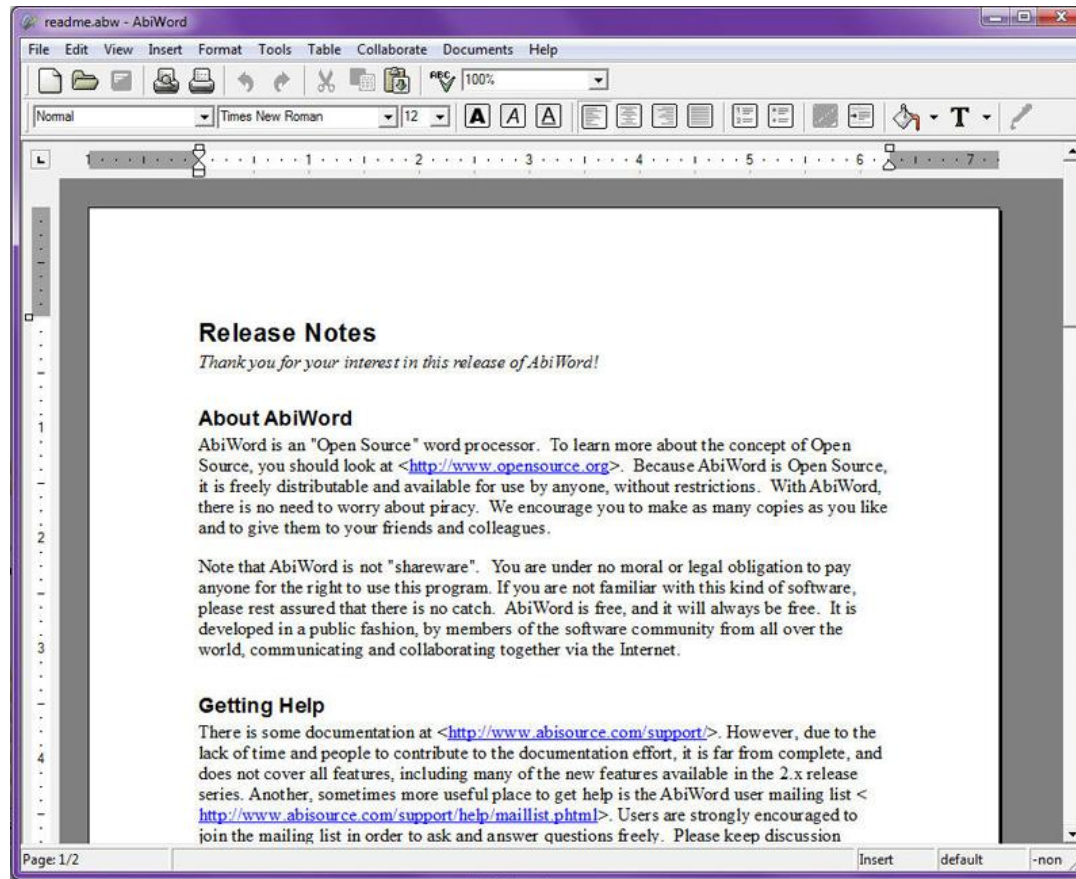
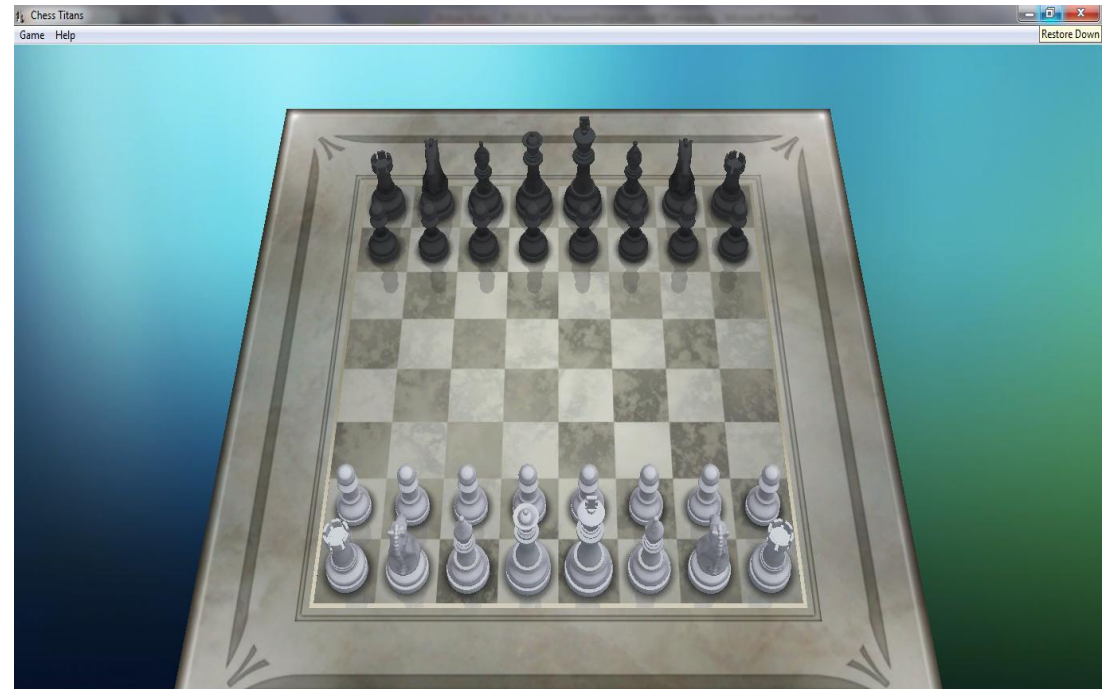


Illustration of an application which uses libvorbisfile to play an [Ogg Vorbis](#) media file

General Purpose Software



Special Purpose Software



Bespoke Software

