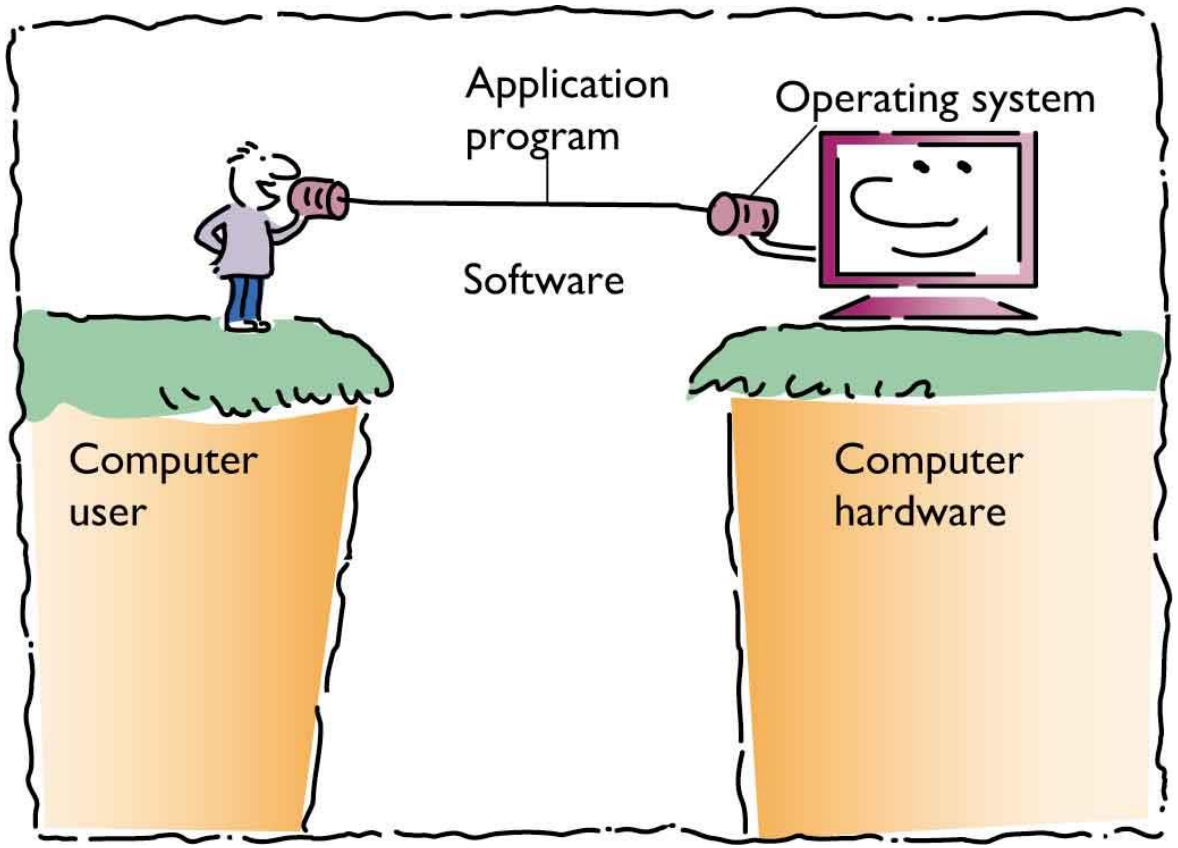
Pengantar Algoritma dan Pemrograman

Tim Penyusun Materi Pengenalan Teknologi Informasi
Institut Teknologi Bandung

Tujuan

- Mahasiswa mampu:
  - Mendeskripsikan proses-proses umum dalam pengembangan perangkat lunak dan program komputer
  - Menjelaskan hubungan antara berpikir komputasional dengan pemrograman
  - Menjelaskan mengapa terdapat berbagai bahasa pemrograman dan memberikan beberapa contoh
  - Mejelaskan perbedaan compiler dengan interpreter
  - Menjelaskan apa itu paradigma pemrograman dan paradigma pemrograman prosedural
  - Menjelaskan struktur dasar program prosedural

User – Software – Hardware

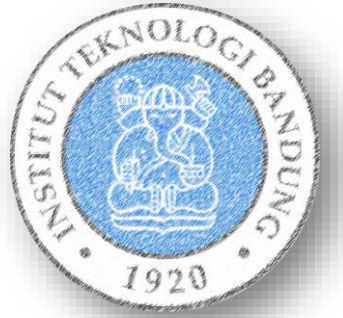


Perangkat Lunak

- **Perangkat Lunak (*software*)** memungkinkan pengguna mengkomunikasikan suatu persoalan kepada komputer dan komputer memberikan solusinya kepada pengguna
  - Tanpa perangkat lunak, komputer hanya mesin bodoh!

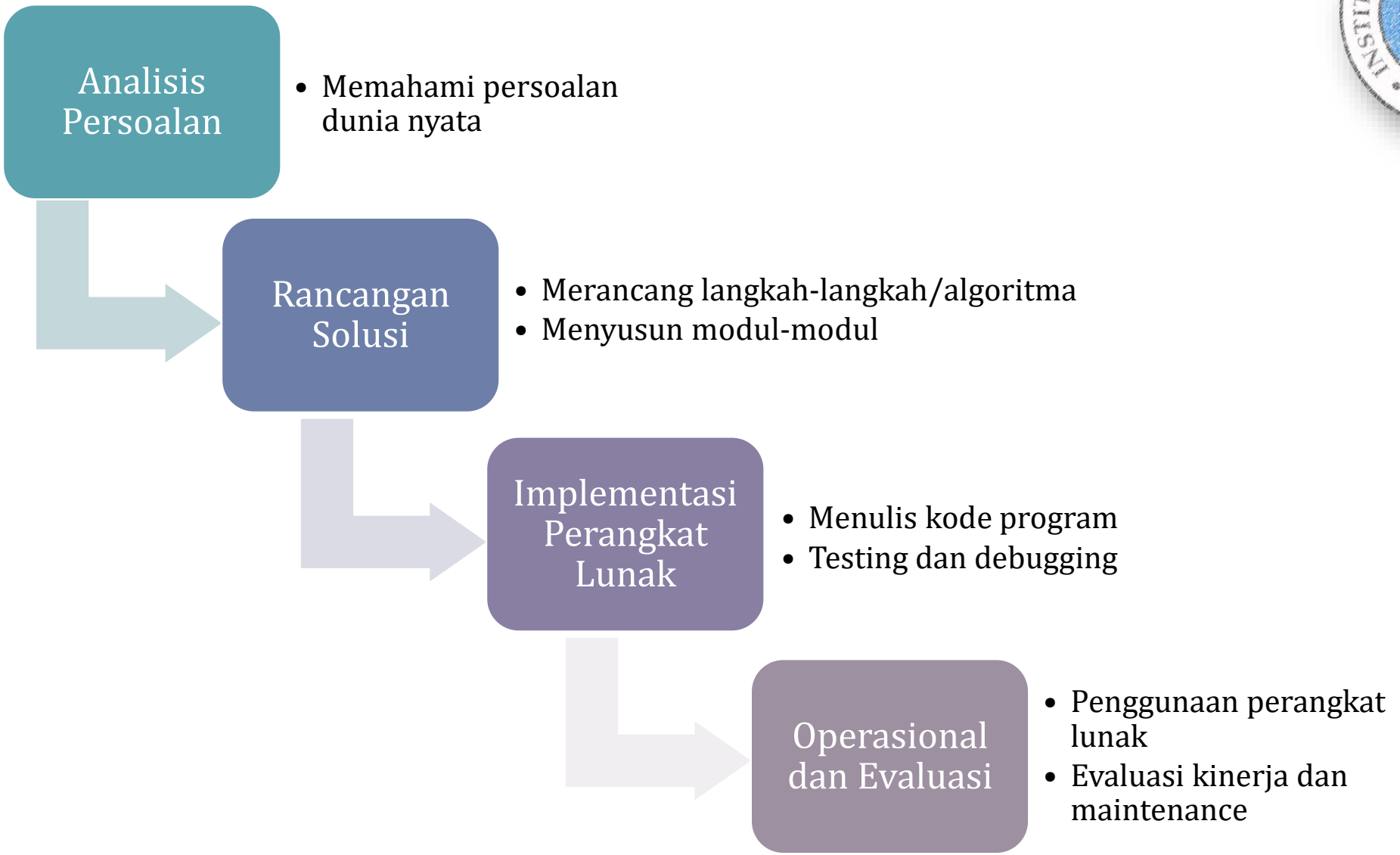
Software = **program** + **data** + dokumentasi

Problem Solving dengan Pemrograman



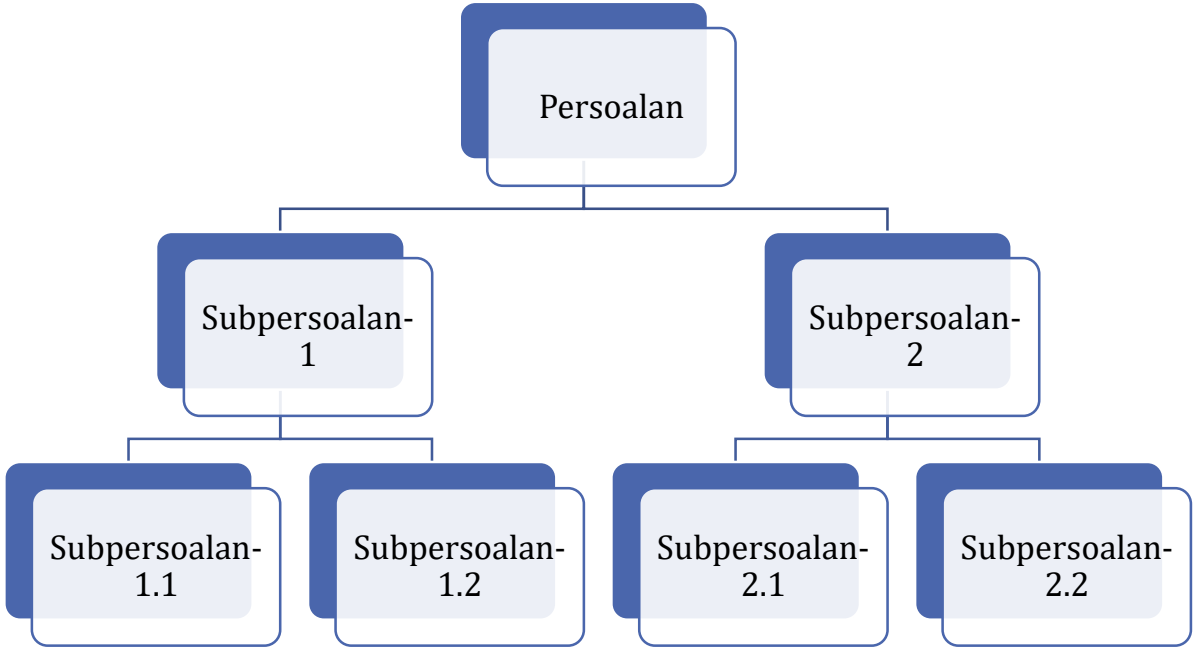
- **Pemrograman (*programming*):** adalah salah satu bentuk penyelesaian persoalan (*problem solving*) di mana persoalan serta solusinya direpresentasikan dalam bentuk yang bisa diproses oleh **komputer**
- Secara umum terdiri atas langkah-langkah sbb.:
  - Memahami persoalan
  - Menyusun rencana untuk menyelesaikan persoalan → **algoritma**
  - Menyusun solusi berdasarkan rencana → **program komputer**
  - Mengevaluasi solusi

Dari Persoalan Menjadi Program (1)



Dari Persoalan Menjadi Program (2)

- Berdasarkan pemahaman terhadap persoalan, *programmer* menyusun daftar persoalan dan mendekomposisi menjadi sub-persoalan
  - Setiap sub-persoalan berpotensi menjadi modul program
  - Dalam kuliah ini, modul program akan disusun dalan **subprogram** (*tunggu beberapa minggu lagi*)
- ***Top down design*: Mulai dari persoalan besar didetilkan sampai pada akhirnya menjadi langkah-langkah penyelesaian sub-persoalan → algoritma**



Diskusi

- Persoalan apa saja dalam bidang studi Anda yang pemecahannya membutuhkan program komputer?

Kemampuan Berpikir Komputasional

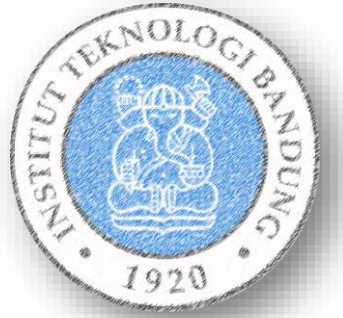
- Memecahkan masalah dengan pemrograman membutuhkan kemampuan berpikir komputasional (*computational thinking*)
- Teknik-teknik berpikir komputasional:
  - Dekomposisi persoalan (*problem decomposition)*
  - Pengenalan pola (*pattern recognition*)
  - Generalisasi pola dan abstraksi (*pattern generalization and abstraction*)
  - Rancangan algoritma (*algorithm design*)
  - Analisis data dan visualisasi

Dari Ide Menjadi Algoritma

- **Algoritma**: himpunan prosedur langkah per langkah untuk menyelesaikan suatu [sub]persoalan
- Dapat ditulis dengan menggunakan teks atau gambar:
  - **Pseudocode** (contoh: notasi algoritmik) → teks; persilangan antara bahasa manusia dan bahasa pemrograman
  - **Flowchart** → diagram
- Algoritma disusun dengan memanfaatkan *control structure* yang menentukan bagaimana urutan langkah dieksekusi
  - *Sequence*: langkah-langkah yang dieksekusi berurutan
  - *Conditional* (percabangan): pilihan langkah
  - *Repetition/loop* (pengulangan): pengulangan langkah

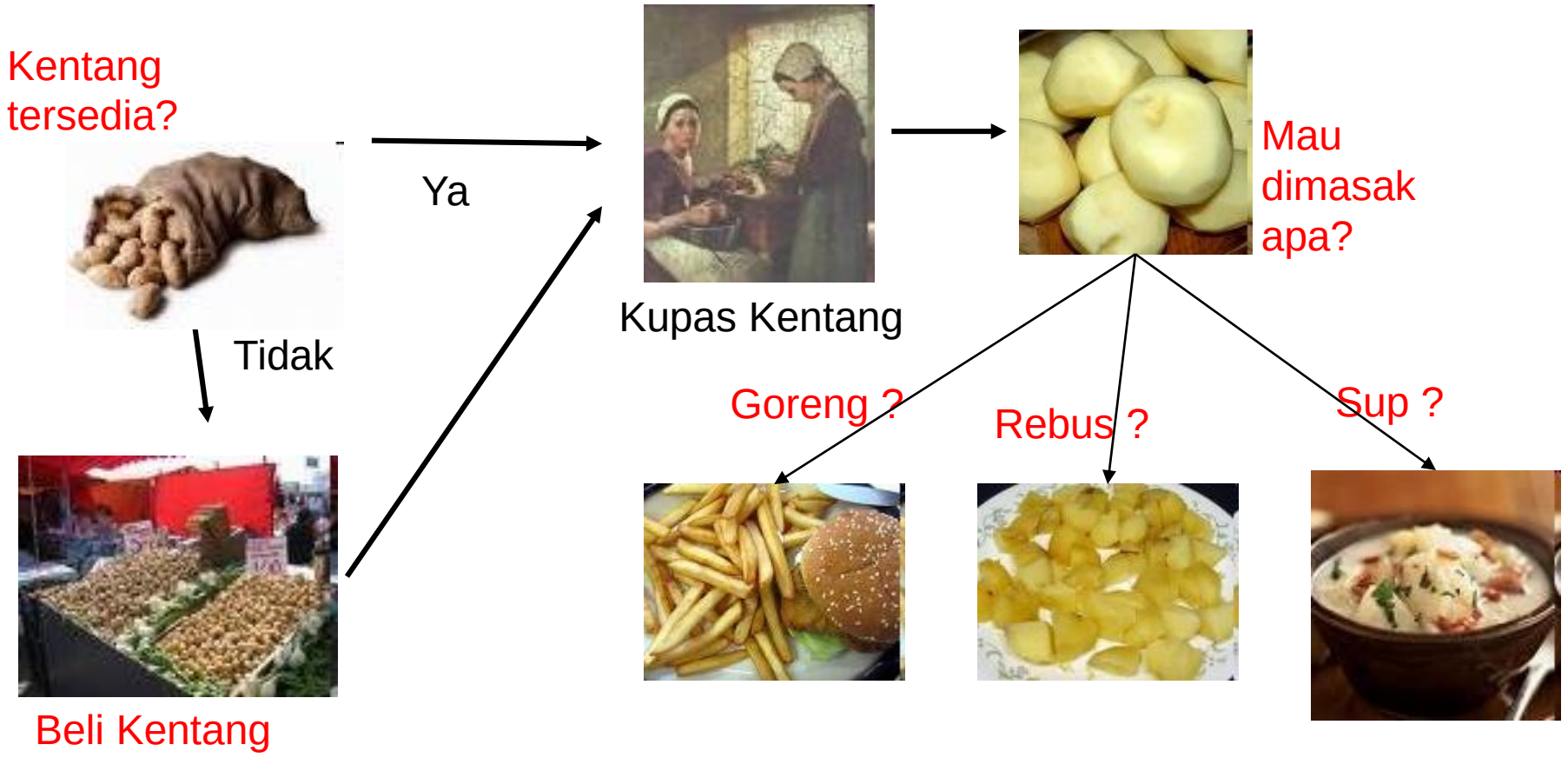
  (*tunggu beberapa minggu lagi*)

Contoh-1:
Memasak Kentang untuk Makan Malam



- Untuk makan malam, sejumlah kentang harus dikupas dan dimasak
- **Keadaan awal**: kantong kentang tersedia di dapur
  - Belum jelas apakah kentang tersedia cukup atau tidak
- **Keadaan akhir**: masakan dengan bahan dasar kentang tersedia dan siap dihidangkan untuk makan malam

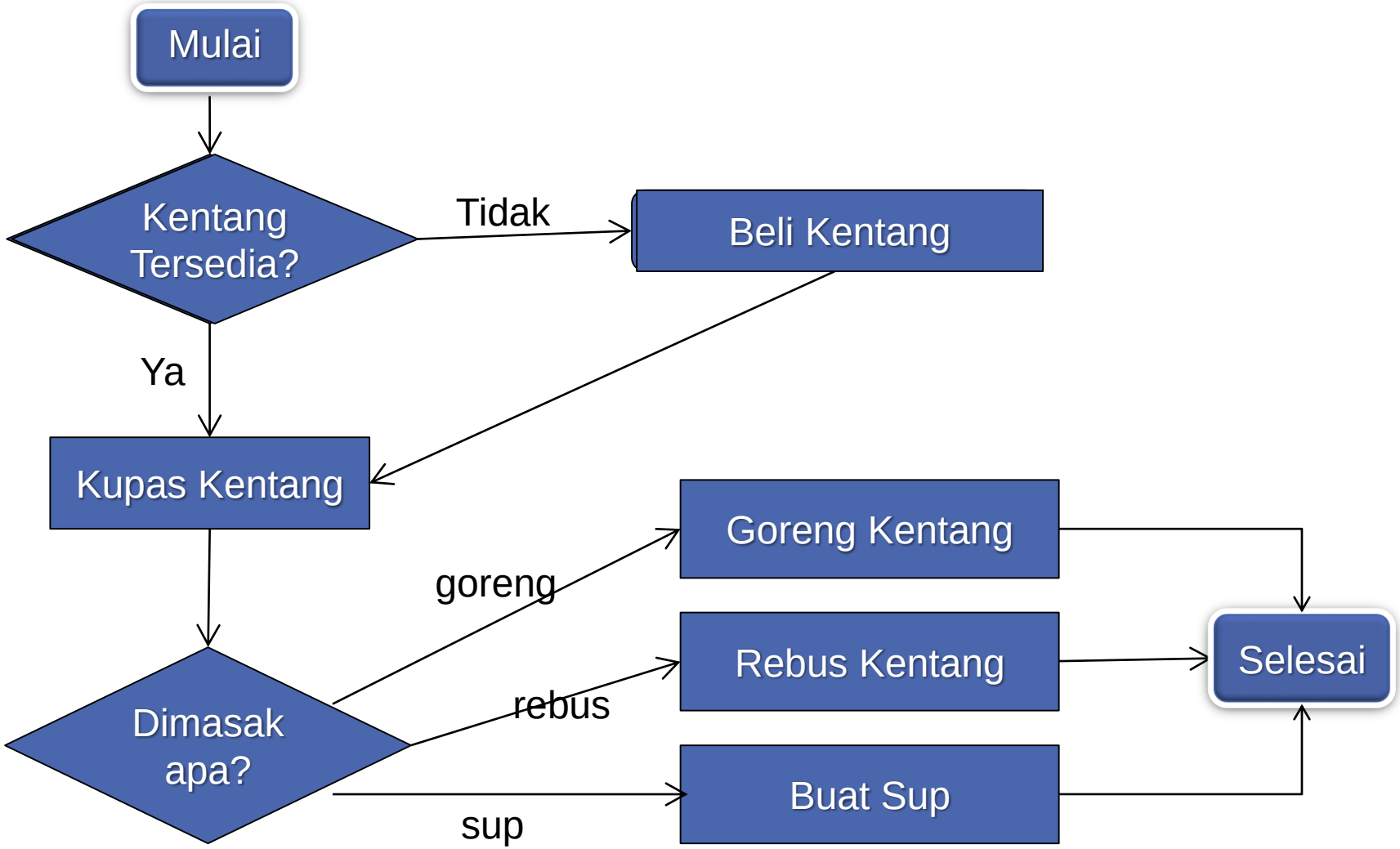
Memasak kentang untuk makan malam



Memasak Kentang untuk Makan Malam - Pseudocode

```
if kentang_tersedia? = tidak then
      Beli_Kentang
{ Di titik ini kentang sudah tersedia }
Kupas_Kentang
if pilihan_masakan = goreng then
      Goreng_Kentang
else if pilihan_masakan = rebus then
      Rebus_Kentang
else { pilihan_masakan = sup }
      Buat_Sup
```

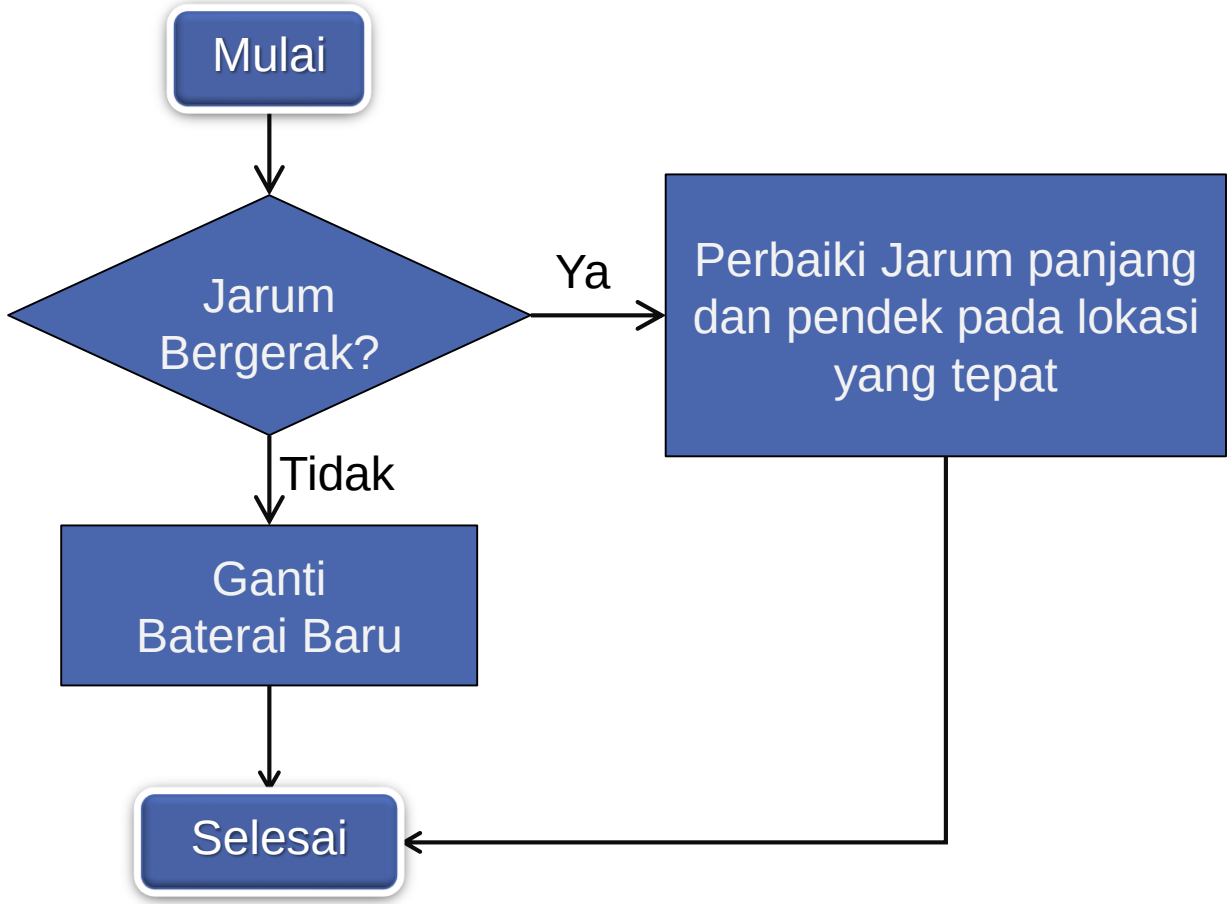
Memasak Kentang untuk Makan Malam - Flowchart



Contoh-2:
Perbaikan Jam Dinding

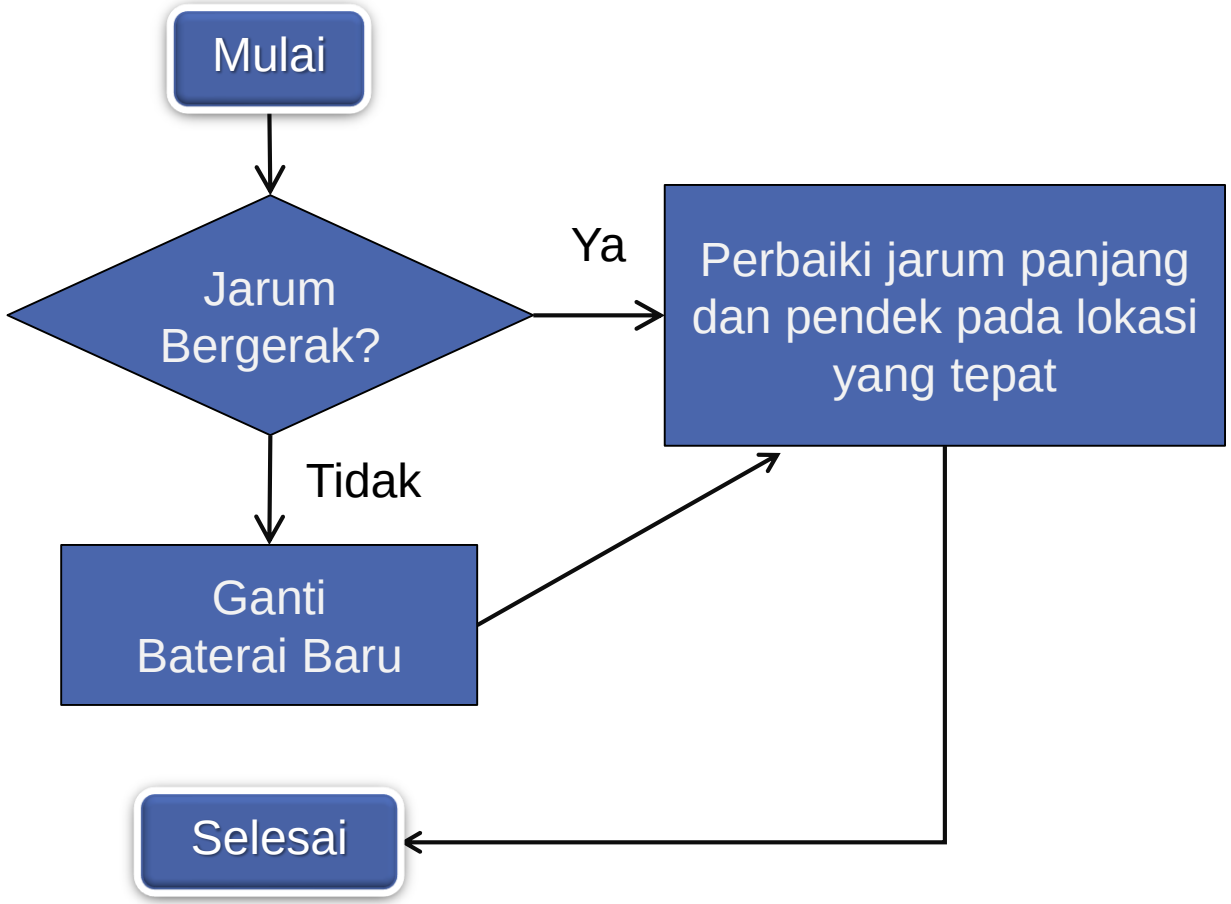
- **Keadaan awal**: Jam dinding tidak menunjukkan waktu yang tepat
- **Keadaan akhir**: Jam dinding menunjukkan waktu yang tepat
- Bila jarum tidak bergerak, ganti baterai
- Jika bergerak berarti baterai masih hidup tinggal dilakukan perbaikan letak jarum jam

Perbaikan Jam Dinding: Flowchart-1



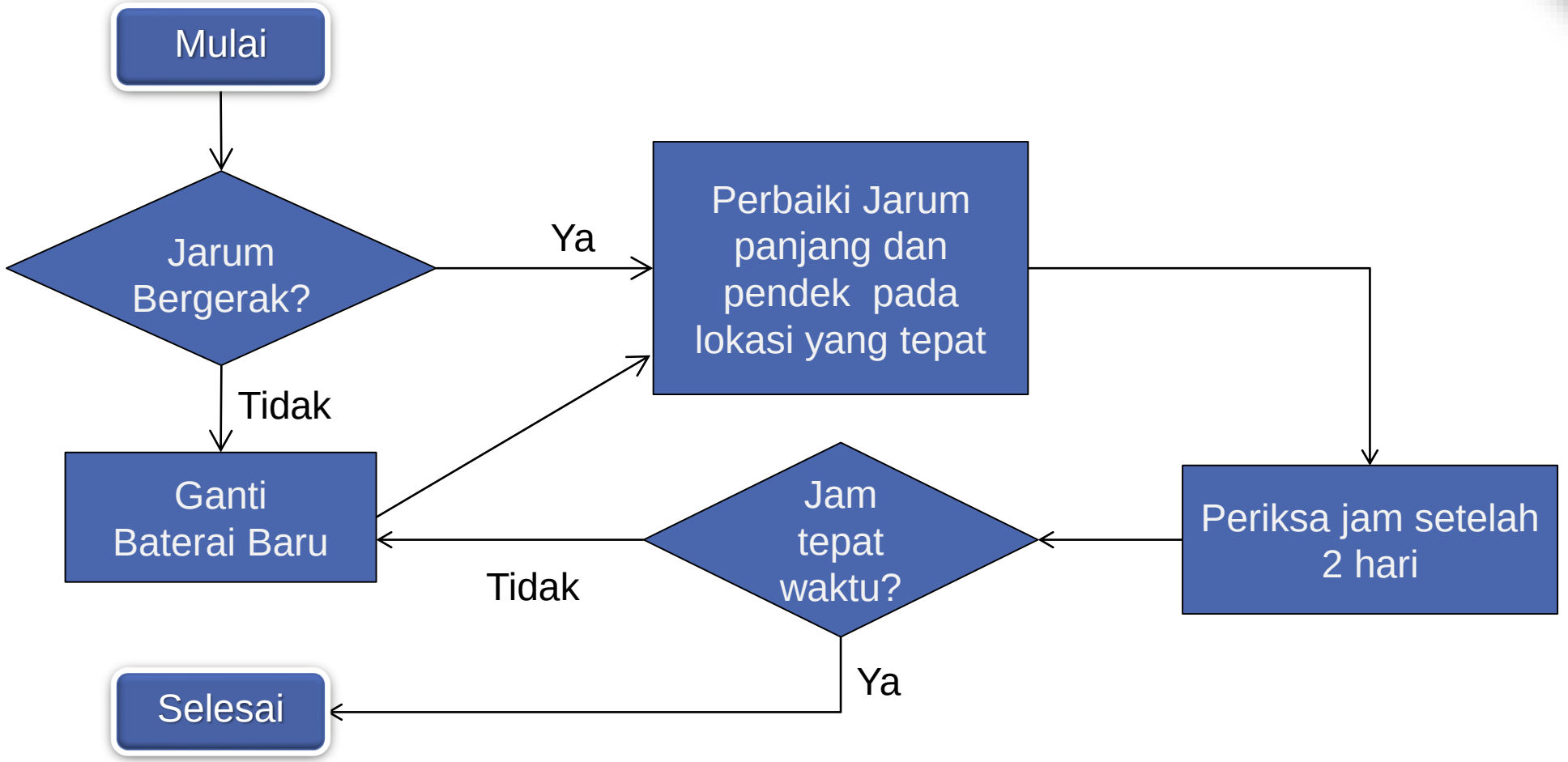
Apa yang salah dengan solusi ini??

Perbaikan Jam Dinding: Flowchart-2



Bagaimana jika ternyata setelah dua hari jam kembali tidak tepat?

Perbaikan Jam Dinding: Flowchart-3



Perbaikan Jam Dinding: Pseudocode

Setara dengan Flowchart-3

```
if jarum_bergerak? = tidak then
      Ganti_Baterai_Baru
{ Di titik ini jarum jam sudah pasti bergerak }
repeat
      Perbaiki_Letak_Jarum_Jam
      Periksa_Jam_Setelah_2_Hari
      if jarum_jam_tepat? = tidak then
            Ganti_Baterai_Baru
until (jarum_jam_tepat? = ya)
```

Contoh-3
Which photo do you want?

2014-JP-03

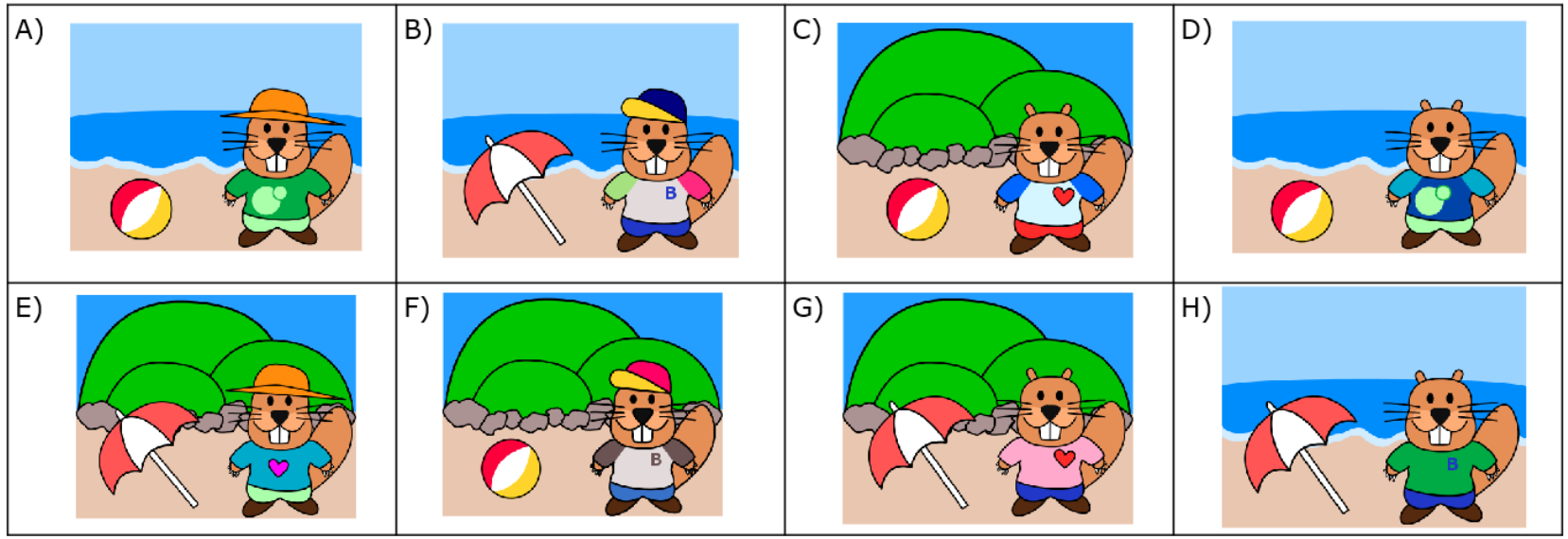
Johnny has 8 photos. He would like to give one of them to Bella. He asks her some questions to find out which photo she wants:

"Do you want a photo with a beach umbrella?" "Yes."

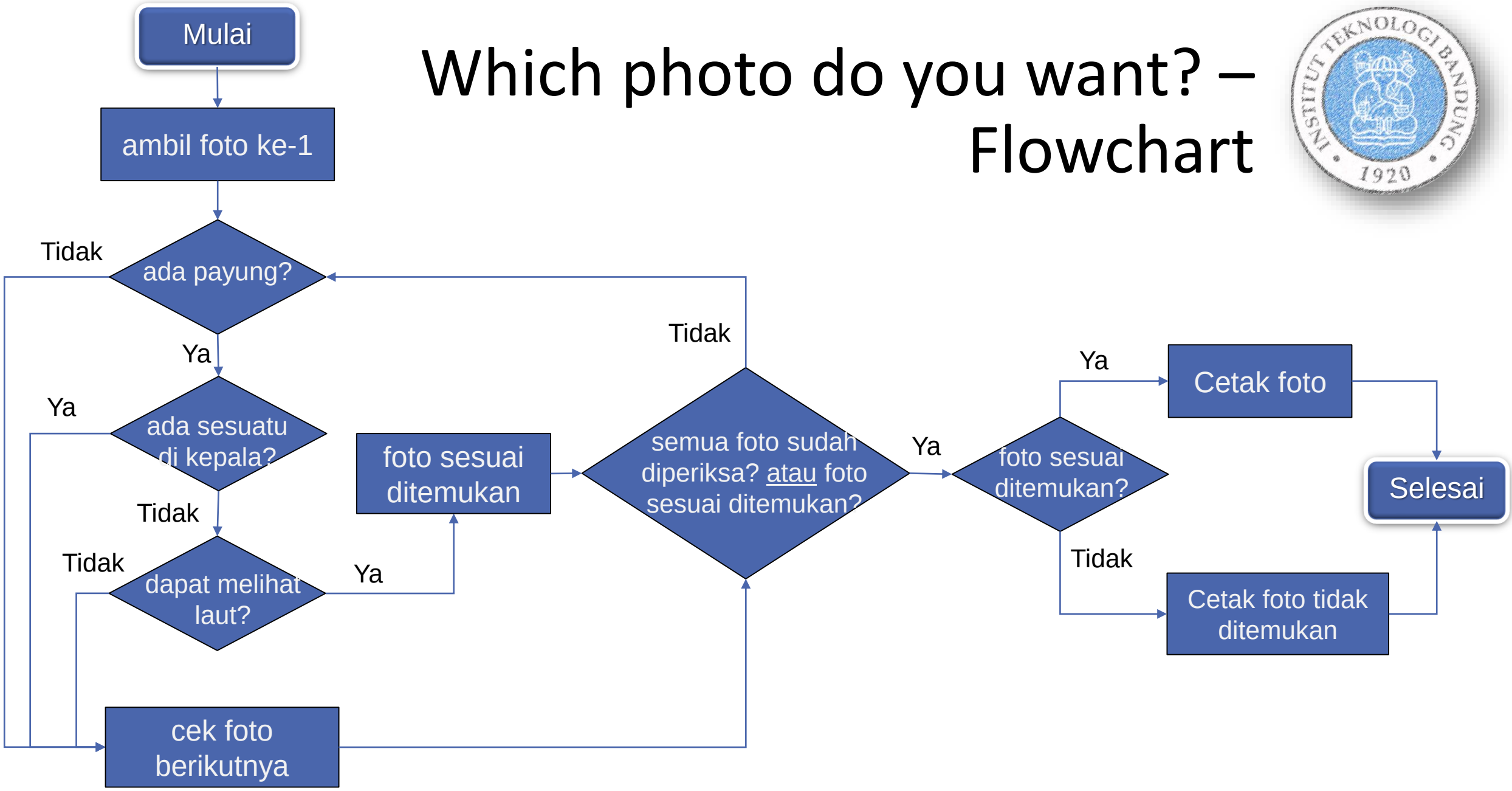
"Do you want a photo where I have something on my head?" "No."

"Do you want a photo where you can see the sea?" "Yes."

Which photo should Johnny give to Bella? (Source: Bebras Challenge)



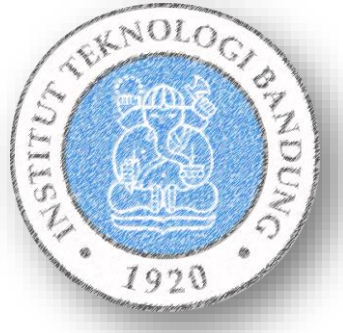
Which photo do you want? – Flowchart



Which photo do you want? – Pseudocode

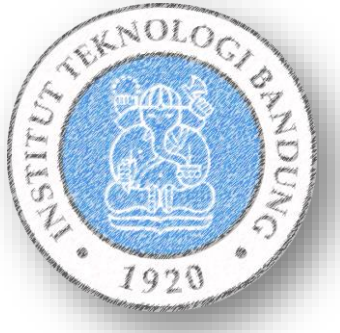
```
ambil_foto_ke_1
repeat
   if ada_payung? = ya then
       if ada_sesuatu_di_kepala? = tidak then
         if dapat_melihat_laut? = ya then
             foto_sesuai_ditemukan
         else
             cek_foto_berikutnya
     else
         cek_foto_berikutnya
   else
       cek_foto_berikutnya
until (semua_foto_sudah_diperiksa) or
      (foto_sesuai_ditemukan)

if (foto_sesuai_ditemukan) then
    cetak_foto
else
    cetak_foto_tidak_ditemukan
```

Dari Algoritma Menjadi Program



- *Programmer* mengubah **algoritma** menjadi **kode program** komputer dengan menggunakan **bahasa pemrograman**
- Proses untuk menuliskan kode program berdasarkan algoritma disebut sebagai ***coding***
  - File hasil menuliskan kode program: ***source code*** (kode sumber)
- Setiap pernyataan dalam algoritma ditranslasikan secara detil ke dalam kode program
- ***Compiler/interpreter*** akan mentranslasi kode program dalam bahasa pemrograman tertentu menjadi bentuk yang dipahami oleh komputer

Bahasa Pemrograman



- Setiap komputer memproses instruksi dalam **bahasa mesin** (*machine language*)
  - Kode-kode numerik yang digunakan untuk mengerjakan operasi-operasi dasar:
    - *Adding and subtracting numbers*
    - *Comparing numbers*
    - *Moving numbers*
    - *Repeating instructions*
- Programmer menggunakan **bahasa pemrograman tingkat tinggi** (high-level languages) untuk menuliskan kode program
  - Pascal, C/C++, Matlab, Python, Fortran, Basic, Java, dll.

Compiler vs Interpreter

- **Compiler**: membaca seluruh kode program sekaligus dan menerjemahkannya menjadi kode yang dipahami mesin komputer

  - Contoh: C/C++, Pascal, Fortran
- **Interpreter**: membaca baris kode satu per satu

  - Contoh: MATLAB, Python

Paradigma Pemrograman

- **Paradigma pemrograman** adalah sudut pandang penyelesaian persoalan dengan program komputer
- Contoh paradigma pemrograman:
  - **Paradigma prosedural (imperatif)** → akan diajarkan di kuliah ini
  - Paradigma berorientasi objek
  - Paradigma deklaratif
  - Dll.
- **Paradigma prosedural (imperatif):** Program didasari oleh **strukturisasi informasi** di dalam memori dan **manipulasi** dari informasi yang disimpan tersebut

Program = Algoritma + Struktur Data

Bahasa Pemrograman Prosedural

- Ada **RIBUAN** bahasa pemrograman di dunia saat ini → termasuk bahasa pemrograman prosedural
- Tidak mungkin semua bahasa pemrograman dipelajari di kuliah
- Oleh karena itu yang diajarkan adalah "belajar pemrograman" → melalui **pola pikir komputasional** dan **paradigma pemrograman prosedural**
- Bahasa pemrograman yang diajarkan di PTI-B: C/C++, Pascal, MATLAB, Python, Fortran
- Di kelas ini akan diajarkan: **Python**

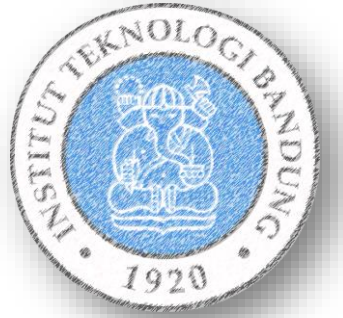
Python

- Bahasa programming tingkat tinggi, direlease oleh Guido van Rossum pada tahun 1991
- Mendukung berbagai paradigma pemrograman. Dalam kuliah ini, hanya akan menggunakan paradigma procedural.
- Interpreter yg tersedia pada beragam sistem operasi:
  - Indentasi untuk menandai blok program
  - case sensitive → perbedaan huruf besar dan kecil berpengaruh
- Python adalah bahasa pemrograman yang ***loosely typed***
  - Tidak perlu mendeklarasikan secara eksplisit tipe data dari variabel

Struktur Dasar Algoritma

Program <JudulProgram> { Spesifikasi Program }
KAMUS { Deklarasi type, variabel, konstanta, fungsi, prosedur }
ALGORITMA { Deretan langkah algoritmik untuk penyelesaian persoalan } { Ditulis dengan pseudocode atau flowchart }

Struktur Dasar Program Python




```
# Program <JudulProgram>
# Spesifikasi Program

# KAMUS
# Penjelasan dalam bentuk komentar
# Deklarasi type, variabel, konstanta, fungsi, prosedur

# ALGORITMA
# Deretan langkah algoritmik untuk penyelesaian  # persoalan
```

Program Pertama

- Buatlah program untuk menuliskan “Hello, World!” ke layar.

print adalah perintah untuk mencetak teks ke layar/monitor

```
# Program HelloWorld
# Mencetak Hello, World! ke layar

# KAMUS
# belum diperlukan

# ALGORITMA
print("Hello, World!")
```