



Pengantar Berpikir Komputasi dan Pemrograman Prosedural



Pengenalan Teknologi Informasi
Institut Teknologi Bandung



Tujuan Kuliah Pemrograman

- Mahasiswa mampu:
 - Menjelaskan bagaimana komputasi/program dimanfaatkan dalam keilmuan fakultas/sekolah
 - Menjelaskan bagaimana proses dari source code menjadi program dengan menggunakan kompilator/interpreter
 - Menjelaskan apa yang dimaksud berpikir komputasi dengan pendekatan prosedural
 - Memahami representasi dan pemrosesan data dan program dalam mesin komputer
 - Membuat program kecil pertama dalam bahasa pemrograman yang dipilih dan memahami aspek eksekusinya”



Apa itu Berpikir Komputasi

- Berpikir menggunakan logika
 - Melakukan sesuatu selangkah demi selangkah
 - Menentukan keputusan bila menghadapi dua kemungkinan yang berbeda
- Salah satu cara untuk memahami dan mengimplementasikan cara berpikir komputasi adalah dengan belajar pemrograman
 - Cara lain?
 - Belajar memahami penggunaan perangkat lunak pengolah kata (contoh: MS Word), pengolah LembarKerja (SpreadSheet, contoh: MS Excel)



Kemampuan Berpikir Komputasi

“Berpikir komputasi memungkinkan kita untuk menggunakan komputasi sesuai dengan kebutuhan kita. Kemampuan ini akan menjadi salah satu kemampuan dasar yang harus dimiliki di abad 21”

(<http://link.cs.cmu.edu/article.php?a=600>)

Kemampuan Dasar yang sudah ada sebelumnya:

- Kemampuan Membaca
- Kemampuan Menulis
- Kemampuan Berhitung

Berpikir Komputasi

- Melibatkan sekumpulan keahlian dan teknik pemecahan masalah yang biasanya digunakan oleh pengembang perangkat lunak untuk menulis program aplikasi komputer.
- Teknik-teknik yang digunakan
 - Dekomposisi
 - Pengenalan Pola (pattern recognition)
 - Generalisasi Pola dan abstraksi(pattern generalization)
 - Untuk mendefinisikan suatu model
 - Rancangan Algoritma dan analisa data /visualisasi



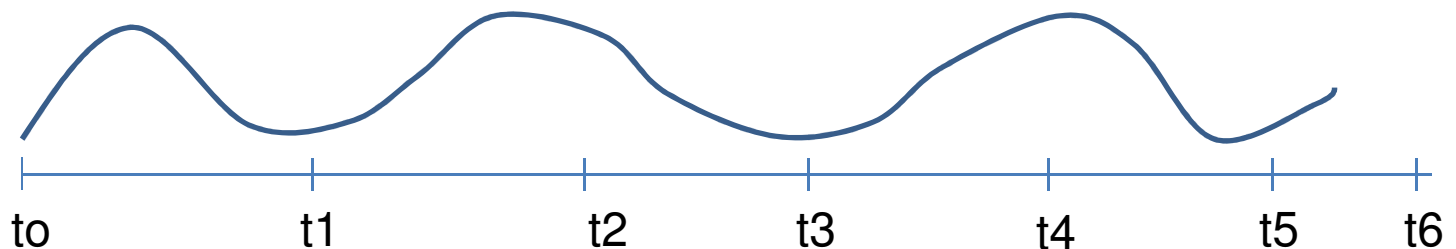
Dekomposisi

- Suatu masalah dipecah menjadi beberapa sub-masalah yang lebih kecil
 - Matematika: $256 = 2 * 100 + 5 * 10 + 6 * 1$
 - Sistem Perpustakaan
 - Peminjaman buku
 - Anggota
 - Dosen
 - Mahasiswa
 - Program komputer: ?



Pengenalan Pola (Pattern Recognition)

- Kemampuan melihat adanya kesamaan yang akan memungkinkan kita untuk melakukan prediksi
 - Pola penjualan saham



- Untuk program komputer, kadang kita bisa menemui pola yang berulang, keadaan ini memungkinkan pemisahan bagian program menjadi procedure/fungsi

Apakah di t6 akan turun atau naik ?



Generalisasi Pola dan Abstraksi

- Kemampuan memilah informasi yang kompleks menjadi lebih sederhana atau membuat informasi lebih bersifat general sehingga memudahkan kita untuk menjelaskan suatu ide
 - Gambar grafik pie-chart untuk abstraksi prosentasi jumlah mahasiswa pria – wanita
 - Lokasi suatu posisi di bumi dapat ditentukan dari kordinat langitude atau latitude
 - Menghitung fibonacci
 - $\text{Fibonacci}[0] = 1$
 - $\text{Fibonacci}[1] = 1$
 - $\text{Fibonnaci}[n] = \text{Fibonacci}[n-1] + \text{Fibonacci}[n-2]$



Rancangan Algoritma

- Kemampuan mengembangkan strategi selangkah demi selangkah untuk pemecahan masalah.
- Rancangan algoritma biasanya dibuat berdasarkan dekomposisi masalah dan identifikasi pola yang akan membantu pemecahan masalah.
 - Urutan memasak kentang
 - Urutan memakai sepatu
 - Implementasi program komputer dalam bahasa C, Pascal, Fortran, dll.

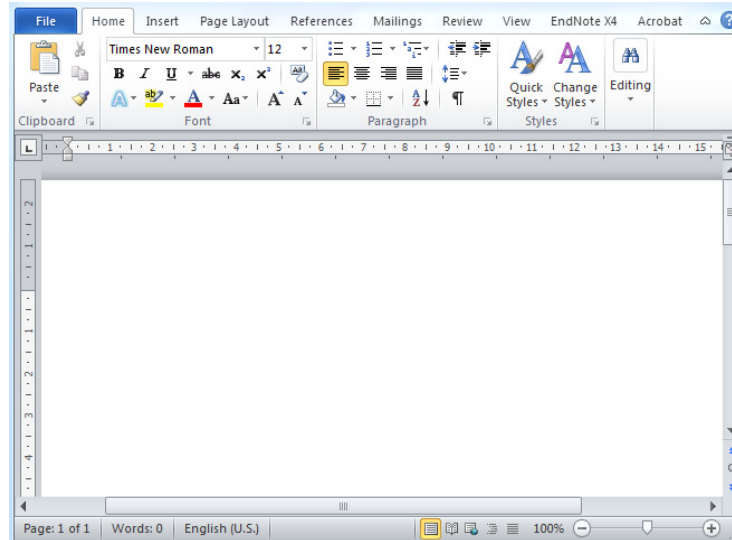


Karakteristik Berpikir Komputasi

- Mampu memberikan **pemecahan masalah** menggunakan **komputer** atau perangkat lain
- Mampu **mengorganisasi** dan **menganalisa data**
- Mampu melakukan representasi data melalui **abstraksi** dengan suatu **model** atau **simulasi**
- Mampu melakukan **otomatisasi solusi** melalui cara berpikir **algoritma** (sekumpulan langkah terurut)
- Mampu melakukan **identifikasi, analisa dan implementasi solusi** dengan berbagai kombinasi langkah/cara dan sumberdaya yang efisien dan efektif
- Mampu melakukan **generalisasi solusi** untuk berbagai masalah berbeda



Pemecahan masalah dengan komputer



Organisasi dan Analisa Data

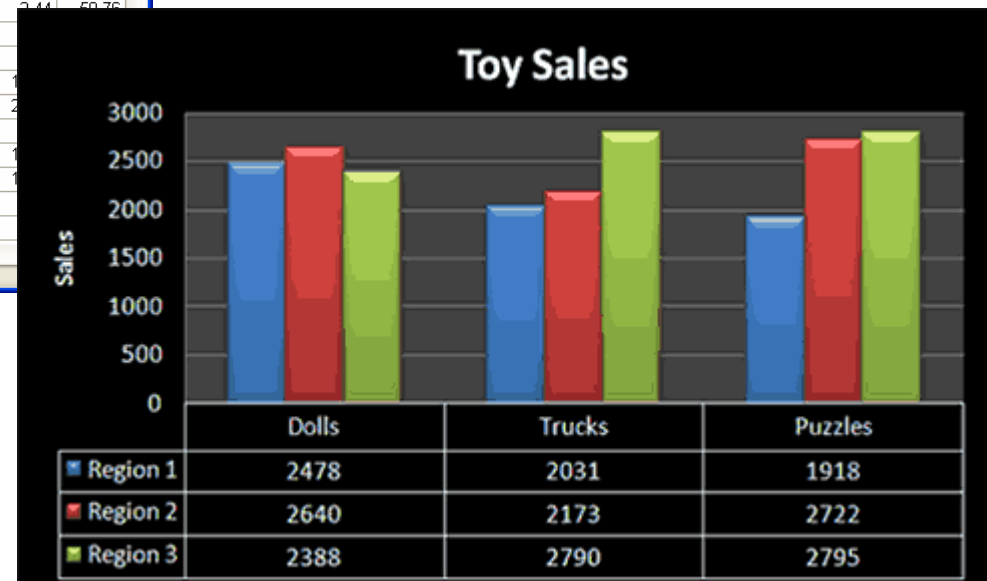
colchester2001.sav [DataSet1] - SPSS Data Editor

File Edit View Data Transform Analyze Graphs Utilities Window Help

1 : outputarea 22UGGE0001 Visible: 29 of 29

	outputarea	ward	v1	v2	v3	v4	v5	v6	v7	v8	v9
1	22UGGE0001	22UGG	9.59	14.38	35.27	19.18	5.14	51.85	2.78	5.50	65.17
2	22UGGE0002	22UGG	3.43	11.21	27.73	32.09	11.53	34.43	4.10	6.84	91.56
3	22UGGE0003	22UGG	5.56	11.40	29.53	26.02	8.19	48.25	2.63	15.57	79.24
4	22UGGE0004	22UGG	1.16	14.34	21.71	25.97	27.91	11.43	37.86	10.37	53.49
5	22UGGE0005	22UGG	3.75	14.33	31.74	24.57	14.68	64.06	26.56	2.54	69.28
6	22UGGE0006	22UGG	4.04	19.88	31.99	17.70	11.49	17.09	3.42	10.00	75.16
7	22UGGE0007	22UGG	6.33	11.71	26.58	25.00	15.19	13.22	22.31	12.28	66.46
8	22UGGE0008	22UGG	4.18	13.09	34.18	12.36	10.55	35.71	.00	7.50	39.27
9	22UGGE0009	22UGG	2.67	5.35	27.27	9.63	13.37	.00	65.00	25.71	35.64
10	22UGGE0010	22UGG	6.92	12.69	33.08	18.85	15.00	10.32	50.79	2.44	59.76
11	22UGGE0011	22UGG	7.96	12.74	24.84	25.16	14.97	37.10	25.81		
12	22UGGE0012	22UGG	9.60	18.36	40.96	19.21	4.52	48.82	.00		
13	22UGGE0013	22UGG	6.69	18.06	30.77	21.74	13.38	42.45	6.60		
14	22UGGE0014	22UGG	5.11	13.92	25.57	29.55	10.80	23.62	.00		
15	22UGGE0015	22UGG	5.83	20.86	26.99	16.26	19.33	31.71	2.44		
16	22UGGE0016	22UGG	7.35	19.17	34.19	23.32	4.79	35.78	.00		
17	22UGGE0017	22UGG	4.42	17.98	26.18	22.08	21.77	31.30	2.29		
18	22UGGE0018	22UGG	4.87	18.18	37.99	19.81	6.17	53.78	2.52		
19	22UGGE0019	22UGG	8.20	16.02	28.52	21.09	16.80	29.46	53.57		

Data View Variable View / SPSS Processor is ready



Contoh Analisa Persoalan: Mengupas Kentang

- Bila ingin makan kentang, tentunya kita perlu memiliki kentang terlebih dahulu
- Jika belum ada, maka beli kentang dulu
- Jika sudah ada maka kentang perlu di kupas
- Setelah dikupas, kita harus memilih, apakah kita mau menggoreng kentang, merebus kentang atau membuat sup.



Mengupas Kentang

Kentang tersedia ?



Ya

Tidak



Beli kentang ?



Kupas Kentang



Mau dimasak apa?

Goreng ?



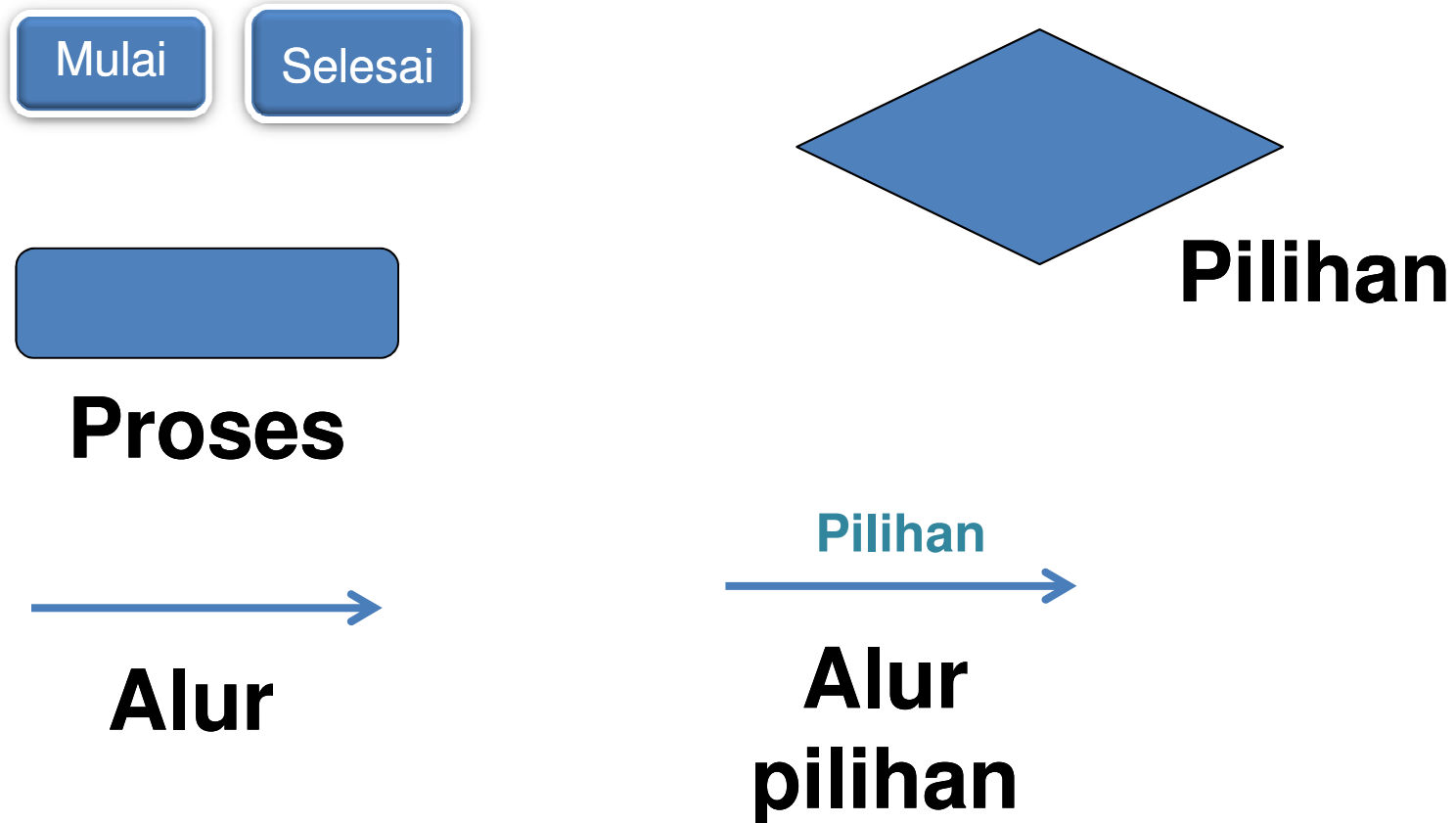
Rebus ?



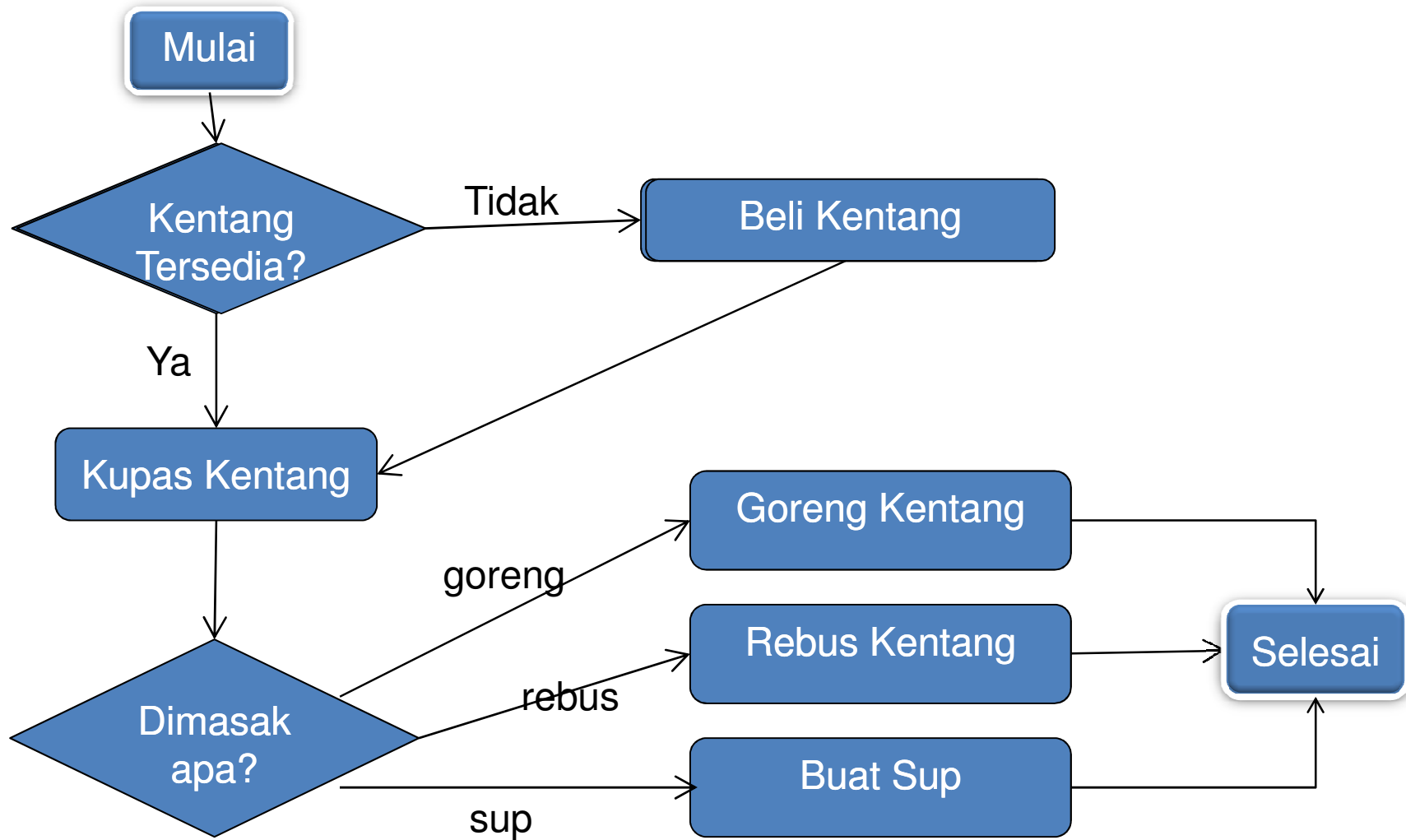
Sup ?



Pengenalan Flowchart (Diagram Alir)



Flow Chart Mengupas Kentang



Contoh Analisa Persoalan: Jarum Jam dinding yang tidak tepat

- Bila jarum tidak bergerak, ganti battery
- Jika bergerak berarti battery masih hidup tinggal dilakukan perbaikan letak jarum jam
- Buatlah flowchartnya!



Diagram Alir (flow chart) Perbaiki Waktu di Jam dinding



Apa yang salah
dengan flowchart
ini??



Diagram Alir (flow chart) Perbaiki Waktu di Jam dinding



Bagaimana jika ternyata setelah dua hari jam kembali tidak tepat?

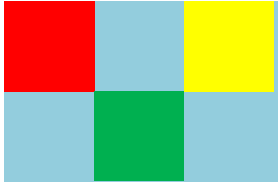
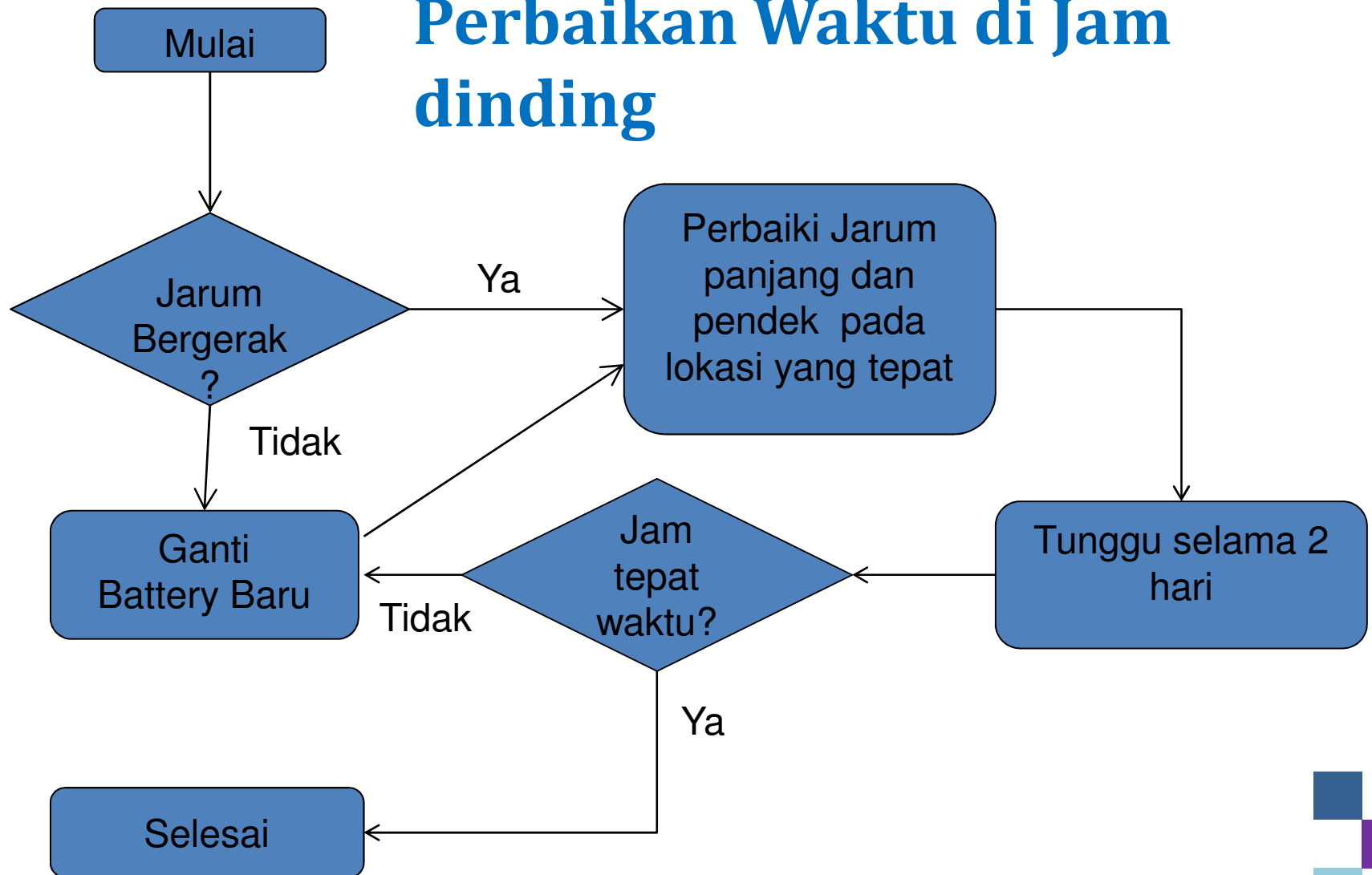
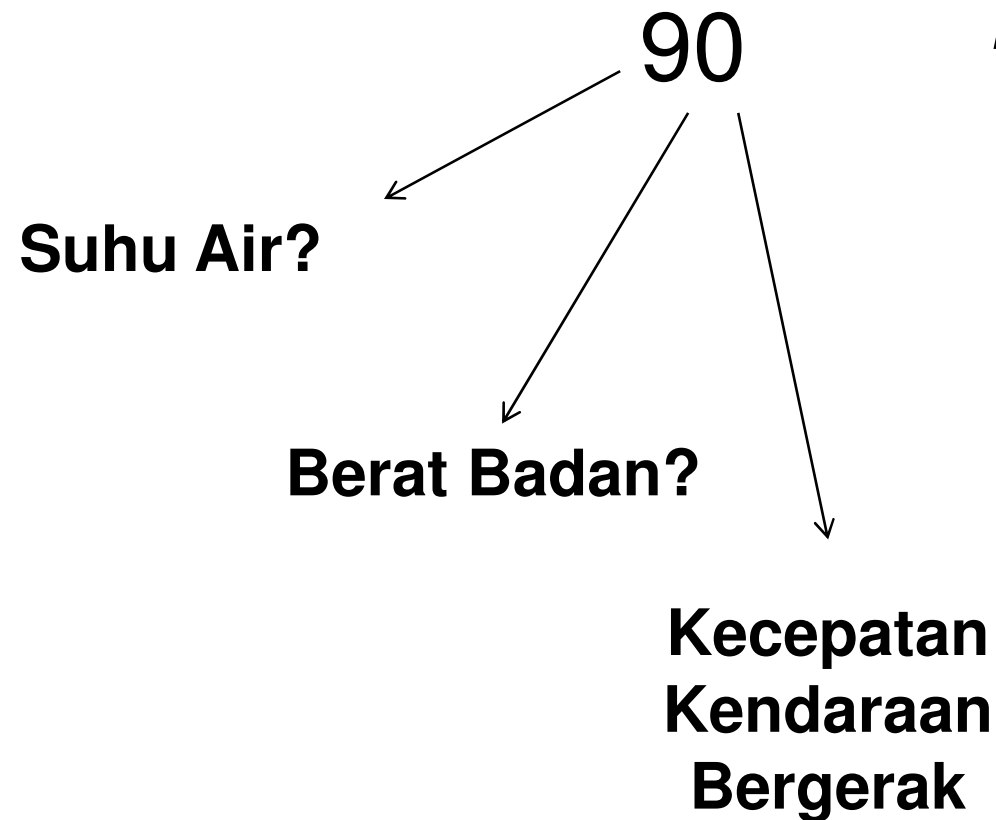


Diagram Alir (flow chart) Perbaiki Waktu di Jam dinding

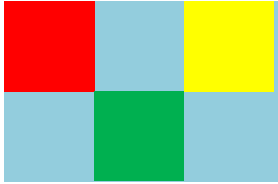


Abstraksi Data



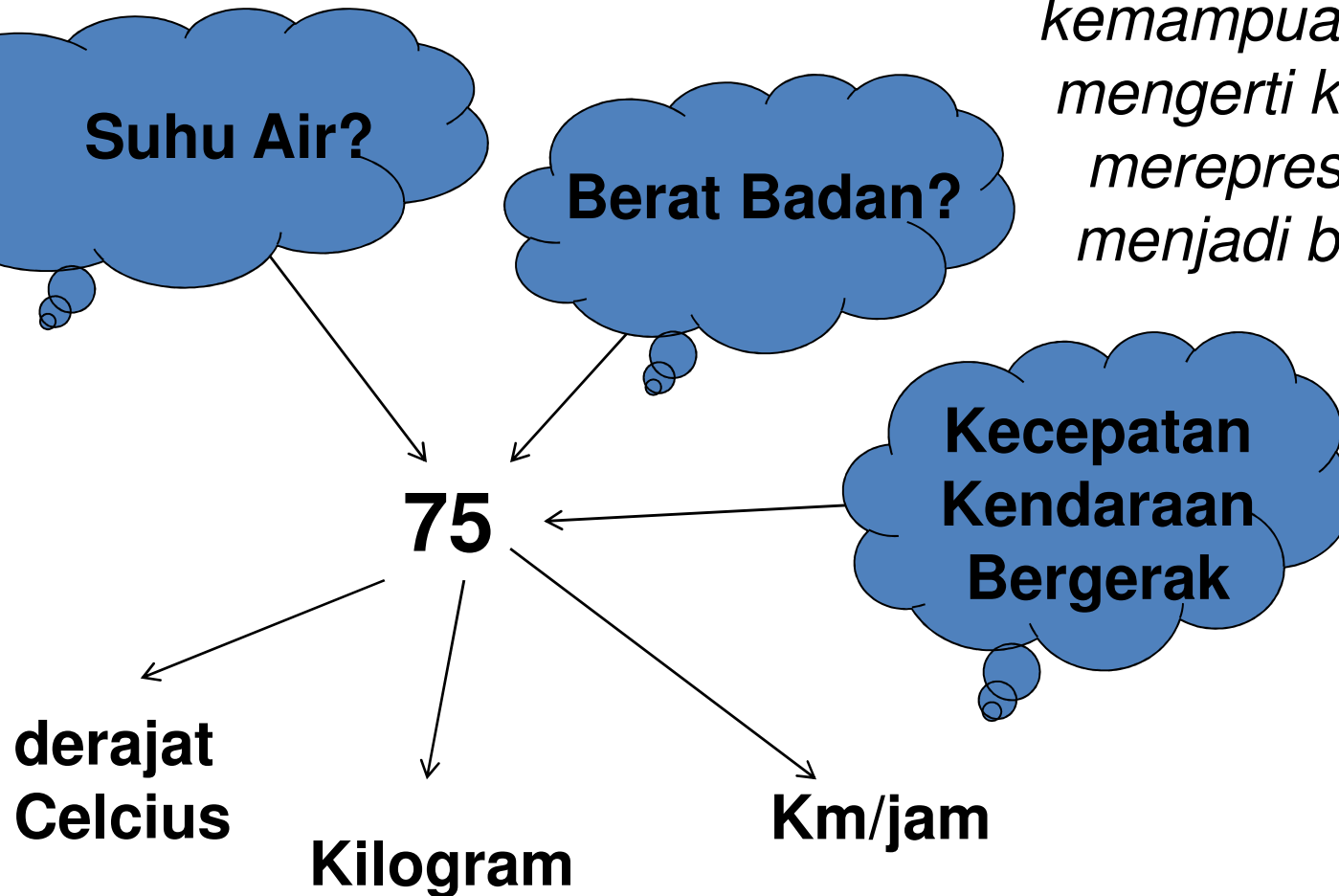
*kemampuan kita untuk
menginterpretasikan
suatu data dengan
konteks masalahnya*





Persoalan Abstraksi Data

*kemampuan kita untuk
mengerti konteks dan
merepresentasikan
menjadi bentuk lain.*



Data Mahasiswa dengan Data Penduduk

Struktur Data

NIM:
Nama
Kota Lahir
Tanggal Lahir
Nama Ayah
Nama Ibu

Nomor KTP:
Nama
Kota Lahir
Tanggal Lahir
Nama Ayah
Nama Ibu



Abstraksi data dengan Model dan Simulasi

- Mampu melakukan **otomatisasi solusi** melalui cara berpikir algoritma (sekumpulan langkah terurut)
- Mampu melakukan **identifikasi, analisa dan implementasi solusi** dengan berbagai kombinasi langkah/cara dan sumberdaya yang efisien dan efektif
- Mampu melakukan **generalisasi solusi** untuk berbagai masalah berbeda



Kemampuan minimum yang diharapkan

- Kemampuan melakukan dekomposisi masalah
 - Menganalisa resep suatu masakan
- Kemampuan mengenali pola
 - Misalnya pola harga barang yang naik kalau setiap lebaran
- Kemampuan menggeneralisir pola dan mengabstraksi pola
 - Abstraksi kalender kerja yang merepresentasikan apa yang dikerjakan dan kapan
- Kemampuan perancangan program
 - Instruksi untuk memasak dengan suatu resep
 - Instruksi membuka paket lemari knock down



Keuntungan berpikir komputasi

- **Percaya diri** dalam berhadapan dengan kompleksitas masalah yang lebih besar dan lebih sulit
- Toleransi terhadap **ambiguitas**
- Mampu berhubungan dengan masalah yang open-ended
- Mampu berkomunikasi dan bekerjasama untuk mencapai tujuan yang sama.



Berpikir Prosedural

- Sejumlah aksi dijalankan secara berurutan (sekuensial)
- Setiap aksi akan memberikan efek eksekusi tertentu
- Jika diikuti terus menerus, aksi-aksi ini harus selesai
 - Tidak bisa terus menerus



Pemrograman Prosedural

- Pemrograman Prosedural (Imperative)
 - Hasil eksekusi program berdasarkan hasil dekomposisi “aksional”.
 - Setiap aksi ini akan dijalankan secara berurutan (sekuensial)
- Pemrograman Non Prosedural
 - Tidak berdasarkan urutan sekuensial
 - Contoh:
 - Pemrograman Deklaratif,
 - Pemrograman Fungsional



Program = Algoritma + Struktur Data



Program Menghitung Tabungan

- Masalah:
 - Tabungan di bank selalu bertambah setiap tahun
 - Bank memiliki bunga yang setiap tahun bertambah
 - BNI memiliki bunga tahunan 10% per tahun
 - Untuk uang 100 ribu, maka setelah satu tahun akan menjadi 110 ribu.
 - Buatlah program yang menghitung uang kita setelah satu tahun.
 - Program akan menanyakan uang kita saat ini, kemudian program akan menampilkan uang kita setelah satu tahun

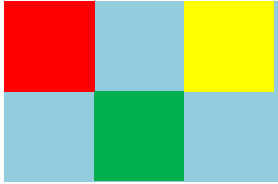


Contoh Eksekusi Program Menghitung Tabungan

Masukkan uang: 1000

Tahun Depan => 1100





Bagaimana Program itu dibuat?



Program Tabungan

input (NilaiRp)

NilaiRp \leftarrow NilaiRp + NilaiRp * 10%

output(NilaiRp)

atau

output (“Masukkan Uang: “)

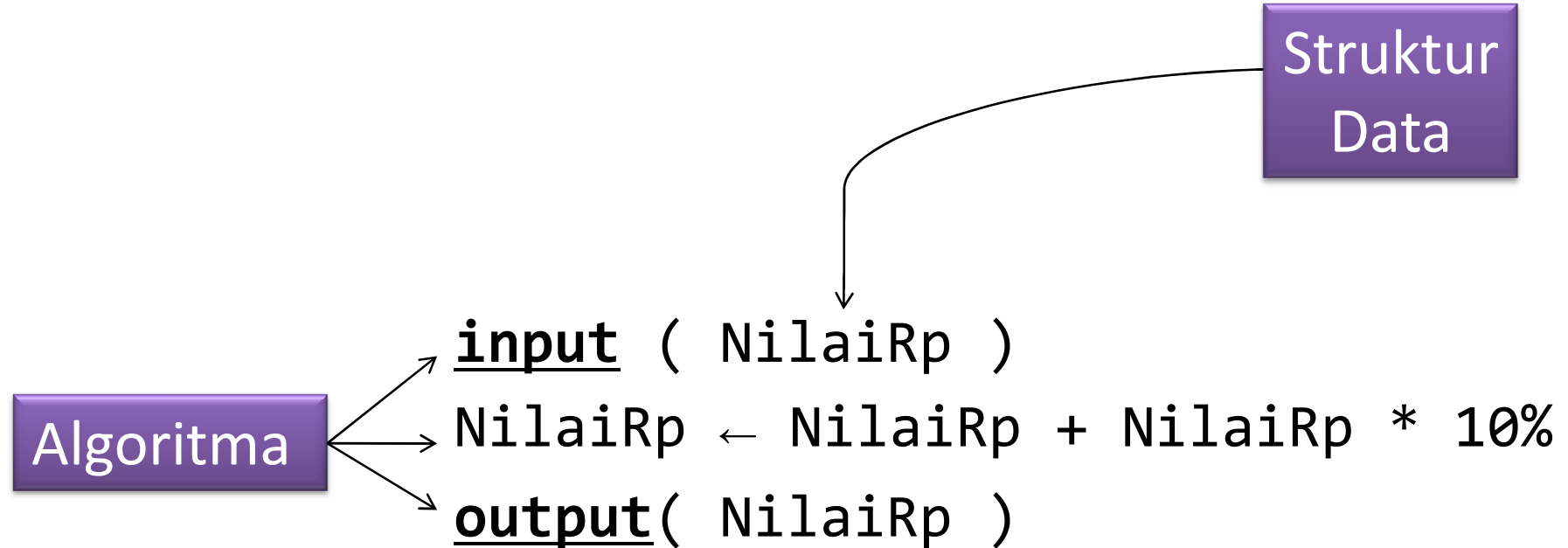
input (NilaiRp)

NilaiRp \leftarrow NilaiRp + NilaiRp * 10%

output(“Tahun Depan => “)

output(NilaiRp)

Program = Algoritma + Struktur Data



Kode Program Bahasa C++

input (NilaiRp)

NilaiRp \leftarrow NilaiRp + NilaiRp * 10%

output(NilaiRp)



*cin: Console Input
(diketikkan lewat
keyboard)*

`cin >> NilaiRp;`

`NilaiRp = NilaiRp + NilaiRp * 0.1;`

`cout << NilaiRp;`

cout: Console Output



Kode Program Bahasa Pascal

input (NilaiRp)

NilaiRp \leftarrow NilaiRp + NilaiRp * 10%

output(NilaiRp)



*readln akan
membaca dari hasil
ketik di keyboard*

→ readln(NilaiRp);

NilaiRp := NilaiRp + NilaiRp * 0.1;

writeln(NilaiRp);

*writeln akan menulis
hasil di layar
komputer*



Kode Program Bahasa Fortran

input (NilaiRp)

NilaiRp \leftarrow NilaiRp + NilaiRp * 10%

output(NilaiRp)

Tanda “” mengindikasikan
keluaran/masukan
standard (keyboard/layar)*

*read akan membaca
dari hasil ketik di
keyboard*

read *, NilaiRp

NilaiRp = NilaiRp + NilaiRp * 0.1;

print *, NilaiRp

*print akan menulis hasil di
layar komputer*



Kode C++

```
int main()  
{
```

```
    int NilaiRp;
```

```
    cin >> NilaiRp;
```

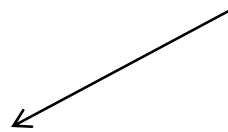
```
    NilaiRp = NilaiRp + NilaiRp * 0.1;
```

```
    cout << NilaiRp;
```

```
    return 0;
```

```
}
```

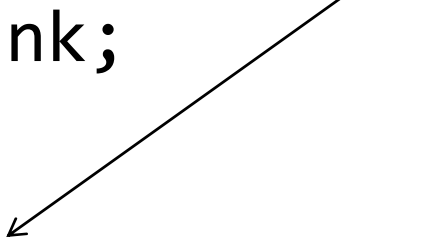
Pendefinisian Data



Kode Pascal

Pendefinisian Data

```
Program HitungUangDiBank;  
var  
    NilaiRp : integer;  
begin  
    readln(NilaiRp);  
    NilaiRp := NilaiRp + NilaiRp * 0.1;  
    writeln(NilaiRp);  
end
```



Kode Fortran

Program HitungUangDiBank

Pendefinisian Data

integer :: NilaiRp

read *, NilaiRp

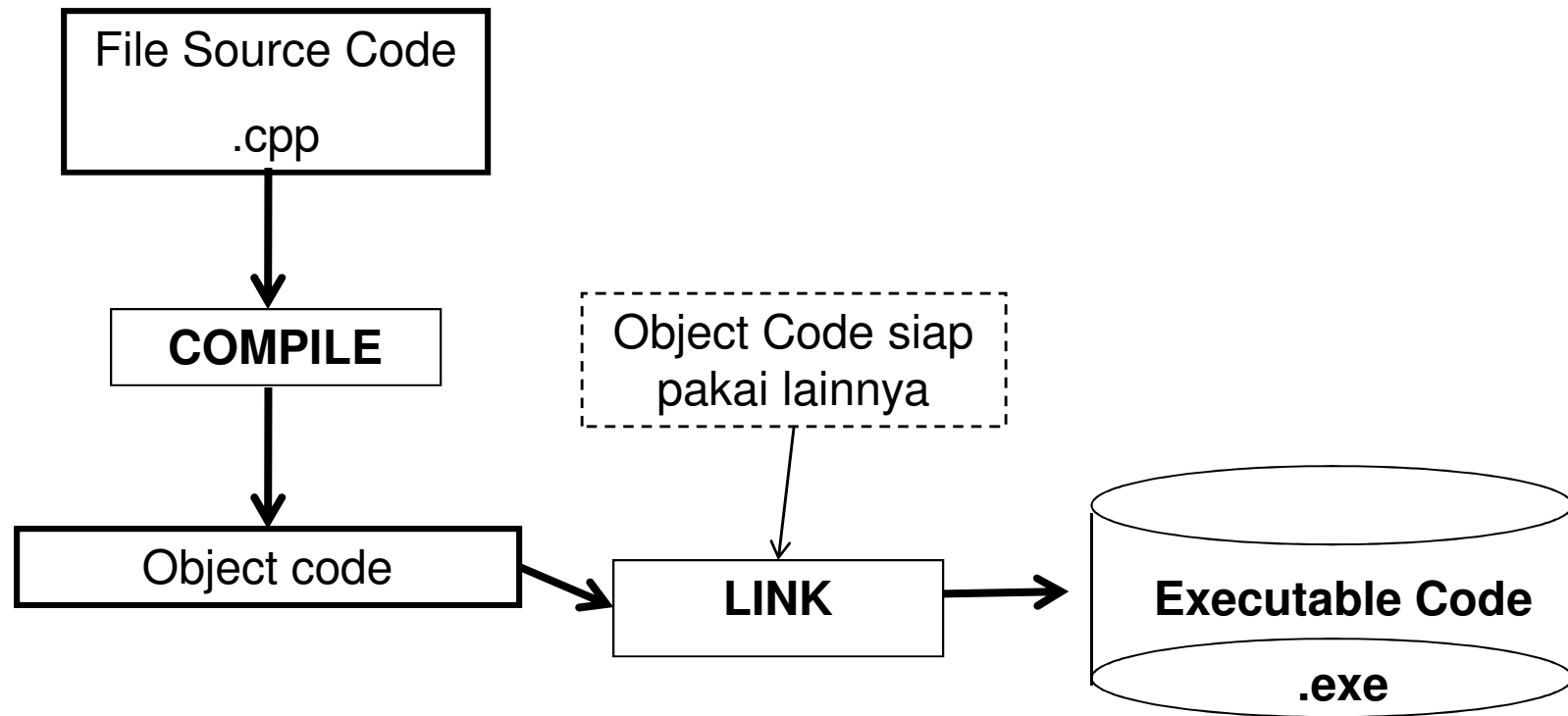
NilaiRp := NilaiRp + NilaiRp * 0.1

print *, NilaiRp

End program HitungUangDiBank



Edit, Kompilasi (Compile) dan Eksekusi



Contoh Edit, Kompilasi (Compile) dan Eksekusi



The screenshot displays the Code::Blocks IDE interface. The main window shows the source code for `main.cpp` in the project `ku1071-bungatabungan`. The code is as follows:

```
1000
1100
Process returned 0 (0x0)   execution time : 5.604 s
Press any key to continue.
```

The output window, titled `C:\Users\bayu\Documents\ProjectC\ku1071-bungatabungan\bin\Debug\ku1071-bungatabungan.exe`, shows the execution results. The status bar at the bottom indicates the current line and column: `Line 3, Column 1`.

Program HitungLuasLingkaran (C++)

```
int main()
{
    /* Kamus */
    float JariJari;
    float Luas;

    /* Algoritma */
    cin >> JariJari;
    Luas = 3.14 * JariJari * JariJari;
    cout << Luas;
}
```





Program HitungLuasLingkaran (Pascal)

```
Program HitungLuasLingkaran;  
(* Kamus *)  
var  
    JariJari : real;  
    Luas      : real;  
begin  
    (* Algoritma *)  
    readln(JariJari);  
    Luas := 3.14 * JariJari * JariJari;  
    writeln(Luas);  
    readln;  
end.
```



Program HitungLuasLingkaran (Fortran)

```
Program HitungLuasLingkaran
```

```
! Kamus
```

```
real :: JariJari
```

```
real :: Luas
```

```
! Algoritma
```

```
read *, JariJari
```

```
Luas = 3.14 * JariJari * JariJari
```

```
print *, Luas
```

```
end program HitungLuasLingkaran
```



Program HitungLuasLingkaran (Fortran) – versi 2

```
Program HitungLuasLingkaran
```

```
! Kamus
```

```
real :: JariJari
```

```
real :: Luas
```

```
! algoritma
```

```
read *, JariJari
```

```
Luas = 3.14 * JariJari ** 2
```

```
print *, Luas
```

```
end program HitungLuasLingkaran
```





Terima Kasih

