

Review Paradigma Pemrograman Prosedural

Tim Pengajar IF1210

Sekolah Teknik Elektro dan Informatika

Ikhtisar

- Review Konsep Dasar Paradigma Prosedural
- Review Notasi Algoritmik dan Bahasa Python 3
- Source Code Standar/Convention

Review Konsep Dasar Pemrograman Prosedural

Review

- Perlunya kemampuan: Computational Thinking
- Konsep dasar berpikir komputasi:
 - Dekomposisi masalah
 - Pengenalan Pola
 - Generalisasi Pola dan Abstraksi
 - Perancangan Algoritma
 - Analisis Data dan Visualisasi
- Memformulasikan persoalan komputasi dan menyelesaikannya dengan bantuan komputer → pemrograman

Pendekatan Formulasi Persoalan & Penyelesaian

- Paradigma: sudut pandang penyelesaian persoalan [dengan program]
- Dalam kuliah IF1210
 - Paradigma Fungsional
 - Rantai Transformasi dari keadaan awal ke keadaan akhir
 - Notasi Fungsional sebagai pengantar → bisa 'diterjemahkan' dalam bahasa lain (Haskell) dengan paradigma yang sama (mempelajari 'tata bahasa' nya)
 - Paradigma Prosedural → review KU1102 + materi tambahan hingga akhir semester

Paradigma Prosedural

- Program didasari oleh **strukturasi informasi** di dalam **memori** dan **manipulasi dari informasi** yang disimpan tersebut
- Hasil eksekusi program berdasarkan hasil **dekomposisi “aksional”**
- Setiap **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

Program = Algoritma + Struktur Data

Notasi Algoritmik

Perlunya Notasi Algoritmik (1)

- Tidak mungkin mempelajari semua bahasa pemrograman → yang penting dipahami: **pola pikir komputasi** dan **paradigma pemrograman**

Belajar memrogram \neq Belajar bahasa pemrograman

- **Notasi algoritmik**: notasi standar yang digunakan untuk menuliskan teks algoritma [dalam paradigma prosedural]
 - **Algoritma** adalah solusi detail secara prosedural dari suatu persoalan dalam notasi algoritmik.
 - **Program** adalah program komputer dalam suatu bahasa pemrograman yang tersedia di dunia nyata.

Perlunya Notasi Algoritmik (2)

- Notasi algoritmik → representasi cara berpikir untuk menyelesaikan persoalan dengan paradigma prosedural
- Membuat program → melihat “kosa kata” translasi notasi algoritmik ke bahasa pemrograman yang dipilih
- Dengan notasi algoritmik yang sama, bisa dibuat program dalam bahasa pemrograman yang berbeda, dengan paradigma pemrograman yang sama
 - Contoh: untuk prosedural: Pascal, Python, C/C++, Fortran

Contoh Kasus: Tabungan

Program = Algoritma + Struktur Data

Notasi Algoritmik

Algoritma

input (NilaiTabungan)
NilaiTabungan \leftarrow NilaiTabungan +
(NilaiTabungan * 0.1)
output(NilaiTabungan)

Struktur
Data

Kode Program Bahasa Python

```
input (NilaiTabungan)  
NilaiTabungan ← NilaiTabungan + (NilaiTabungan * 0.1)  
output(NilaiTabungan)
```

Input

NilaiTabungan = int(input())

NilaiTabungan = NilaiTabungan + NilaiTabungan * 0.1

print(NilaiTabungan)

Output

Kode Program Bahasa Pascal

```
input (NilaiTabungan)  
NilaiTabungan ← NilaiTabungan + (NilaiTabungan * 0.1)  
output(NilaiTabungan)
```

*readln akan
membaca dari hasil
ketik di keyboard*

→ readln(NilaiTabungan);

NilaiTabungan := NilaiTabungan +
NilaiTabungan * 0.1;

*writeln akan menulis
hasil di layar
komputer*

→ writeln(NilaiTabungan);

Kode Program Bahasa C++

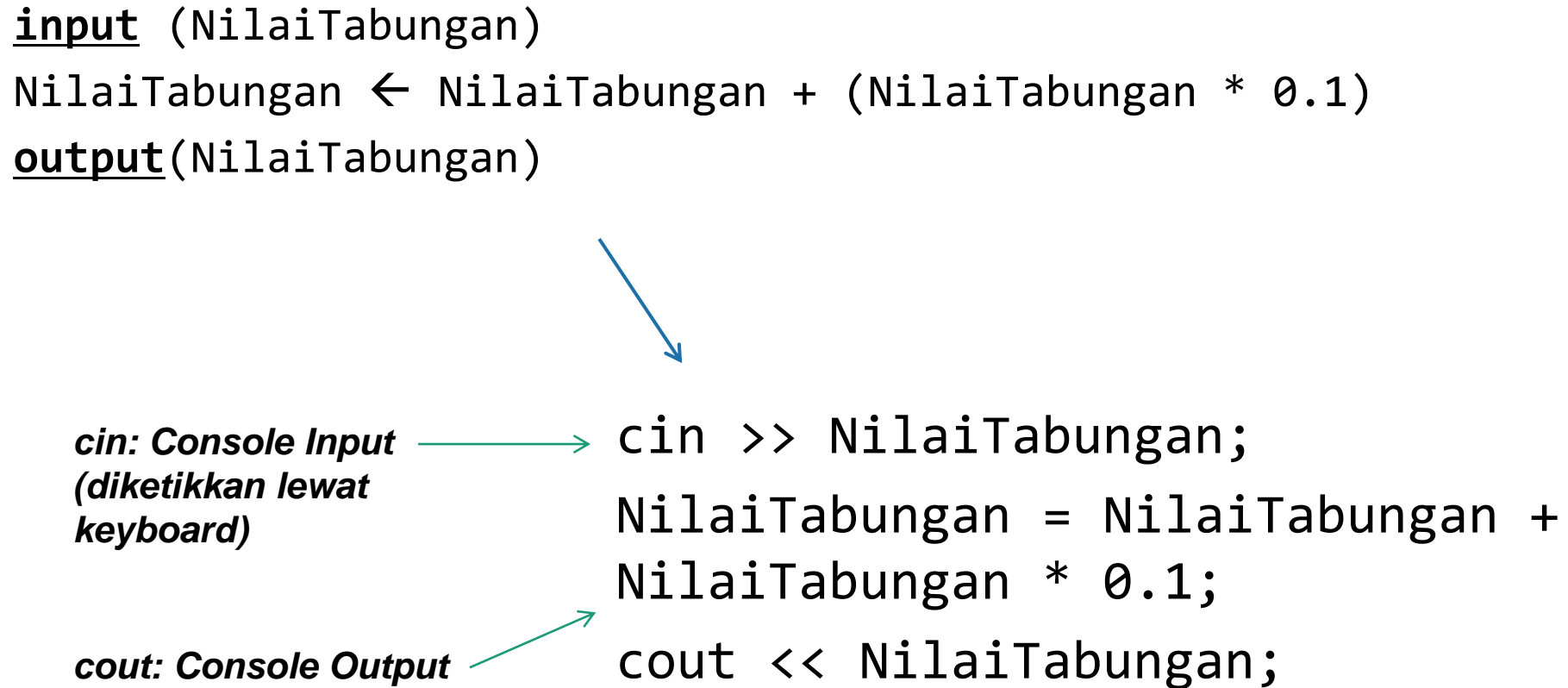
input (NilaiTabungan)

NilaiTabungan \leftarrow NilaiTabungan + (NilaiTabungan * 0.1)

output(NilaiTabungan)

cin: Console Input
(diketikkan lewat
keyboard)

cout: Console Output



```
cin >> NilaiTabungan;  
NilaiTabungan = NilaiTabungan +  
NilaiTabungan * 0.1;  
cout << NilaiTabungan;
```

Kode Program Bahasa Fortran

input (NilaiTabungan)

NilaiTabungan \leftarrow NilaiTabungan + NilaiTabungan * 10%

output(NilaiTabungan)

Tanda '' mengindikasikan
keluaran/masukan
standard (keyboard/layar)*

*read akan membaca
dari hasil ketik di
keyboard*

→ read *, NilaiTabungan

NilaiTabungan = NilaiTabungan +
NilaiTabungan * 0.1;

*print akan menulis hasil di
layar komputer*

→ print *, NilaiTabungan

Struktur Dasar Program Prosedural (Notasi Algoritmik)

Program <JudulProgram>
{ Spesifikasi Program }

KAMUS

{ Deklarasi type, variabel, konstanta, fungsi,
prosedur }

ALGORITMA

{ Deretan langkah algoritmik untuk penyelesaian
persoalan }

{ **Memanfaatkan notasi algoritmik** }

Struktur Dasar Program Python

```
# Program <JudulProgram>  
# Spesifikasi Program  
  
# KAMUS  
# Penjelasan dalam bentuk komentar  
# Deklarasi variabel, konstanta, fungsi, prosedur  
  
# ALGORITMA  
# Deretan langkah algoritmik untuk penyelesaian persoalan
```


Contoh Program (Notasi Algoritmik)

Program Test

{ Spesifikasi Program: menghitung $A + B$ }

KAMUS

{ Deklarasi variabel }
A, B : integer

ALGORITMA

input(A)
input(B)
 $A \leftarrow A + B$
output(A)
output(B)

Contoh Program (Python)

```
# Program Test
# Spesifikasi : Menghitung nilai A dan B

# KAMUS
# A : int
# B : int

# ALGORITMA
A = int(input()) # input
B = int(input())

A = A + B        # proses

print(A)         # output
print(B)
```

Komentar

- Dalam bahasa pemrograman komentar adalah bagian program yang tidak dieksekusi
 - Bagian ini hanya digunakan untuk memberikan penjelasan suatu langkah, rumus, ataupun bisa hanya berupa keterangan

Python	Notasi Algoritmik
# ini komentar	{ ini komentar }

Kamus

- Kamus dipakai untuk mendeklarasi nama-nama yang digunakan dalam program
- Deklarasi nama yang didefinisikan pemrogram
 - type
 - variabel
 - konstanta
- Deklarasi BUKAN instruksi
- Di Python, deklarasi variabel dilakukan bersama dengan inisialisasi nilai
 - Type data ditentukan oleh type nilai inisialisasi
 - Dengan demikian, kamus diisi keterangan tentang variabel apa saja yang digunakan

Python

```
# i : int  
# JumlahUang : float  
# Titik : Point;
```

Notasi Algoritmik

```
i : integer  
JumlahUang : real  
Titik : Point
```

Variabel

- Contoh deklarasi dan inisialisasi variabel:

Python	Notasi Algoritmik
<pre># KAMUS # i : int # A : float</pre>	<pre>KAMUS i : <u>integer</u> A : <u>real</u></pre>
<pre># ALGORITMA i = 100 A = i * 50 ...</pre>	<pre>ALGORITMA i ← 100 A ← i * 50</pre>

Tipe data primitif/Tipe dasar

- Disediakan oleh bahasa pemrograman

Python	Notasi Algoritmik	Domain Nilai
bool	<u>boolean</u>	true; false Python: True, False
int	<u>integer</u>	bilangan bulat negatif, 0, bilangan bulat positif Contoh: 1; -144; 999; 0
float	<u>real</u>	bilangan riil, contoh: 3.14; 4.01E+1
char	<u>character</u>	Pada Notasi Algoritmik karakter/huruf, ditandai dengan kutip tunggal; Contoh: 'A'; '#'; 'b'
string	string	Pada Notasi Algoritmik kumpulan karakter/huruf, ditandai dengan kutip ganda Contoh: "xcxcx"; "AB"

Tipe Data Bentuk (1)

- Tipe data bentukan/komposit / record
 - Tidak tersedia secara otomatis, harus dibuat oleh programmer
 - Dibentuk dari gabungan tipe dasar
 - Dalam notasi algoritmik terdapat sintaks khusus untuk mendeklarasikan type
 - Komponen type dideklarasikan dengan nama khusus dan nama ini digunakan untuk mengakses komponen type
 - Di Python, struktur type bentukan dapat diimplementasikan sebagai tuple
 - Setiap elemen tuple diakses berdasarkan urutan dalam tuple
 - Tuple pada dasarnya adalah struktur yang immutable, tidak bisa mengubah nilai elemen tuple satu per satu, namun harus membentuk nilai baru keseluruhan

Tipe Data Bentukan (2)

	Notasi Algoritmik	Python
Deklarasi type (KAMUS)	<pre> <u>type</u> Point : < x : <u>integer</u>; y : <u>integer</u> > </pre>	<pre> # type Point: # (x:int, # y:int) </pre>
Deklarasi variabel (KAMUS)	<pre> P : Point a, b : <u>integer</u> </pre>	<pre> # P : Point # a, b : int </pre>
Akses elemen (ALGORITMA)	<pre> P.x ← 0 P.y ← 1 a ← P.x b ← P.y + 1 </pre>	<pre> P = (0,1) a = P[0] b = P[1] + 1 # akses elemen menggunakan urutan # kemunculan elemen di tuple # elemen pertama pada urutan ke-0 </pre>

Konstanta

- Berbeda dengan variable, suatu konstanta **tidak boleh diubah** nilainya dalam algoritma
- Contoh deklarasi di KAMUS:

Python	Notasi Algoritmik
PI = 3.14159	<u>constant</u> PI : <u>real</u> = 3.14159
Nilai = 1000	<u>constant</u> Nilai : <u>integer</u> = 1000
NMax = 100	<u>constant</u> NMax : <u>integer</u> = 100

Di Python, tidak ada mekanisme untuk mendeklarasikan variabel sebagai konstanta. Pada prakteknya, biasanya variabel yang ditetapkan sbg konstanta diletakkan di file terpisah, misalnya constant.py dan file tersebut dipanggil dalam program yang menggunakannya.

Assignment

- *Assignment*: Pemberian nilai suatu variabel
 - Di Python, nilai yang diberikan pada suatu variable menentukan type data variabel tersebut
- Ruas kiri harus **variable**
- Ruas kanan harus **ekspresi/nilai/variabel yang sudah jelas nilainya**

Python	Notasi Algoritmik
<code><RuasKiri> = <RuasKanan></code>	<code><RuasKiri> ← <RuasKanan></code>
<u>Contoh:</u> <code>i = 10</code> <code>Nama = "Maya"</code> <code>X = i + 10</code>	<u>Contoh:</u> <code>i ← 10</code> <code>Nama ← "Maya"</code> <code>X ← i + 10</code>

Nilai X di-assign dengan ekspresi

Jenis Ekspresi

	Python	Notasi Algoritmik
Ekspresi Aritmatika: operan numerik (integer/real), hasil: numerik (integer/real)	$A + B$ $x + 2 * y$ $P - 2 * Q + R/S$	$A + B$ $x + 2 * y$ $P - 2 * Q + R/S$
Ekspresi Relasional: operan numerik (integer/real), hasil: boolean	$A < B$ $X == Y$ $Total \geq nilai$	$A < B$ $X = Y$ $Total \geq nilai$
Ekspresi Logika: operan: boolean, hasil : boolean	$A \text{ and } B$ $C \text{ or } B$ $\text{not}(\text{True})$	$A \text{ and } B$ $C \text{ or } B$ not (true)

Operator Tipe Dasar

Type Data	Python	Notasi Algoritmik
Integer	* / + - // % < > <= >= == !=	* / + - <u>div</u> <u>mod</u> < > ≤ ≥ = ≠
Boolean	and or not ==	<u>and</u> <u>or</u> <u>not</u> =
Real	* / + - < > <= >= !=	* / + - < > ≤ ≥ ≠
Character	= !=	= ≠
String	= != +	= ≠ +

Input/Output (1)

- Perintah **input**: pemberian nilai **variabel** dari piranti masukan, misal: keyboard → dibaca atas masukan dari pengguna

Python	Notasi Algoritmik
input	input
<u>Contoh:</u> A=int(input()) # A bertype int B=float(input()) # B bertype float C=input() # C bertype string	<u>Contoh:</u> <u>input</u> (A) <u>input</u> (B) <u>input</u> (C)

Input/Output (2)

- Perintah **output**: penulisan nilai (variabel/konstanta/hasil ekspresi) ke piranti keluaran, misal: monitor/layar

Python	Notasi Algoritmik
print	output
Contoh: <pre>print(A) # menulis A, diakhiri newline print("Hello",end='') #menulis tanpa newline print(A * 4) #menulis hasil perkalian A*4 Print("A = " + str(A)) #menulis A = <nilai A></pre>	Contoh: <pre><u>output</u>(A) <u>output</u>("Hello") <u>output</u>(A*4) <u>output</u>("A = ",A)</pre>

Contoh

- Terjemahkan program Python berikut ke notasi algoritmik

```
# Program Jumlah2Pecahan
# Menghitung pembilang dan penyebut pecahan dari penjumlahan dua
# buah pecahan yang diketahui pembilang dan penyebutnya

# KAMUS
# pembilang1, pembilang2, pembilang3 : int
# penyebut1, penyebut2, penyebut3 : int

# ALGORITMA
pembilang1 = int(input())
penyebut1 = int(input())

pembilang2 = int(input())
penyebut2 = int(input())

pembilang3 = pembilang1 * penyebut2 + pembilang2 * penyebut1
penyebut3 = penyebut2 * penyebut1

print(str(pembilang3) + "/" + str(penyebut3))
```

Jumlah 2 Pecahan (Notasi Algoritmik)

Program JumlahPecahan

{Menghitung pembilang dan penyebut pecahan dari penjumlahan dua buah pecahan yang diketahui pembilang dan penyebutnya}

KAMUS

pembilang1, pembilang2, pembilang3 : integer
penyebut1, penyebut2, penyebut3 : integer

ALGORITMA

input(pembilang1)
input(penyebut1)
input(pembilang2)
input(penyebut2)

$\text{pembilang3} \leftarrow \text{pembilang1} * \text{penyebut2} + \text{pembilang2} * \text{penyebut1}$
 $\text{penyebut3} \leftarrow \text{penyebut1} * \text{penyebut2}$

output(pembilang3, "/", penyebut3)

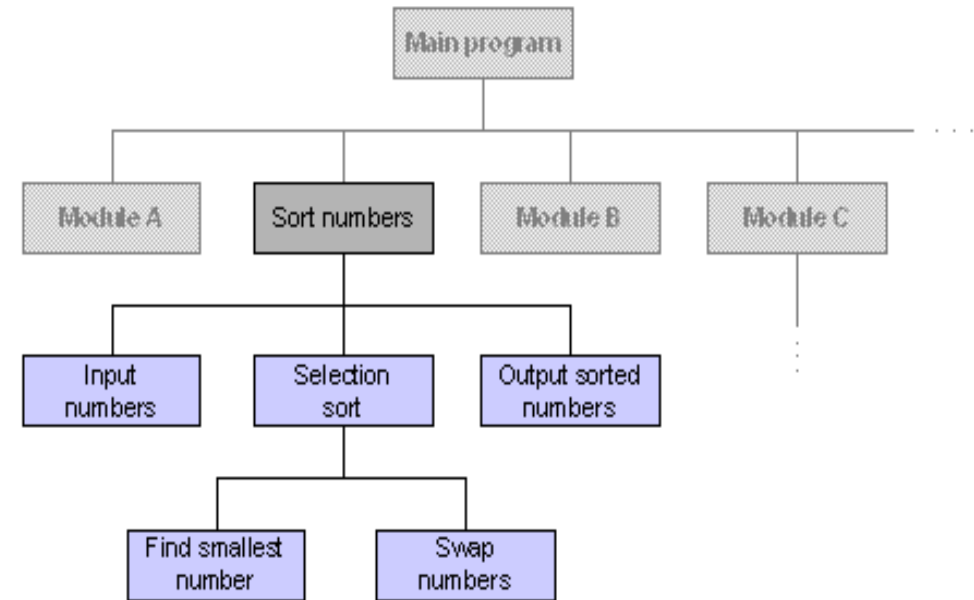
Lebih Lanjut ...

- Notasi algoritmik lebih lanjut dan translasinya di Python akan dipelajari satu per satu dalam materi-materi berikutnya
- Bahan bacaan:
 - Slides “Kumpulan Materi Dasar Pemrograman dengan Python 3” – Bahan Kuliah KU1102/Pengenalan Komputasi
 - Diktat “Dasar Pemrograman Bagian Pemrograman Prosedural” → dengan notasi algoritmik
- Seluruhnya tersedia di situs kuliah online

Source Code Standard/ Convention

Contoh (1)

- Pembuatan software untuk *spreadsheet* atau basis data
- Salah satu modul yang diperlukan:
 - Melakukan pengurutan (*sorting*)



Sumber: <http://courses.cs.vt.edu/csonline/SE/Lessons/Procedural/index.html>

Contoh (2)

- Pembuatan software untuk *spreadsheet* atau basis data
- Tiap modul dibuat oleh orang yang berbeda, masing-masing memiliki 'style'
- Harus ter'integrasi'kan dengan baik → perlu **standar**



Sumber gambar:

<http://www.oncoursesystems.com/school/webpage/11191090/1181281>

Coding Standard

“The best applications are coded properly. This sounds like an obvious statement, but by ‘properly’, I mean that the code not only **does its job well**, but is also **easy to add to, maintain and debug.**”

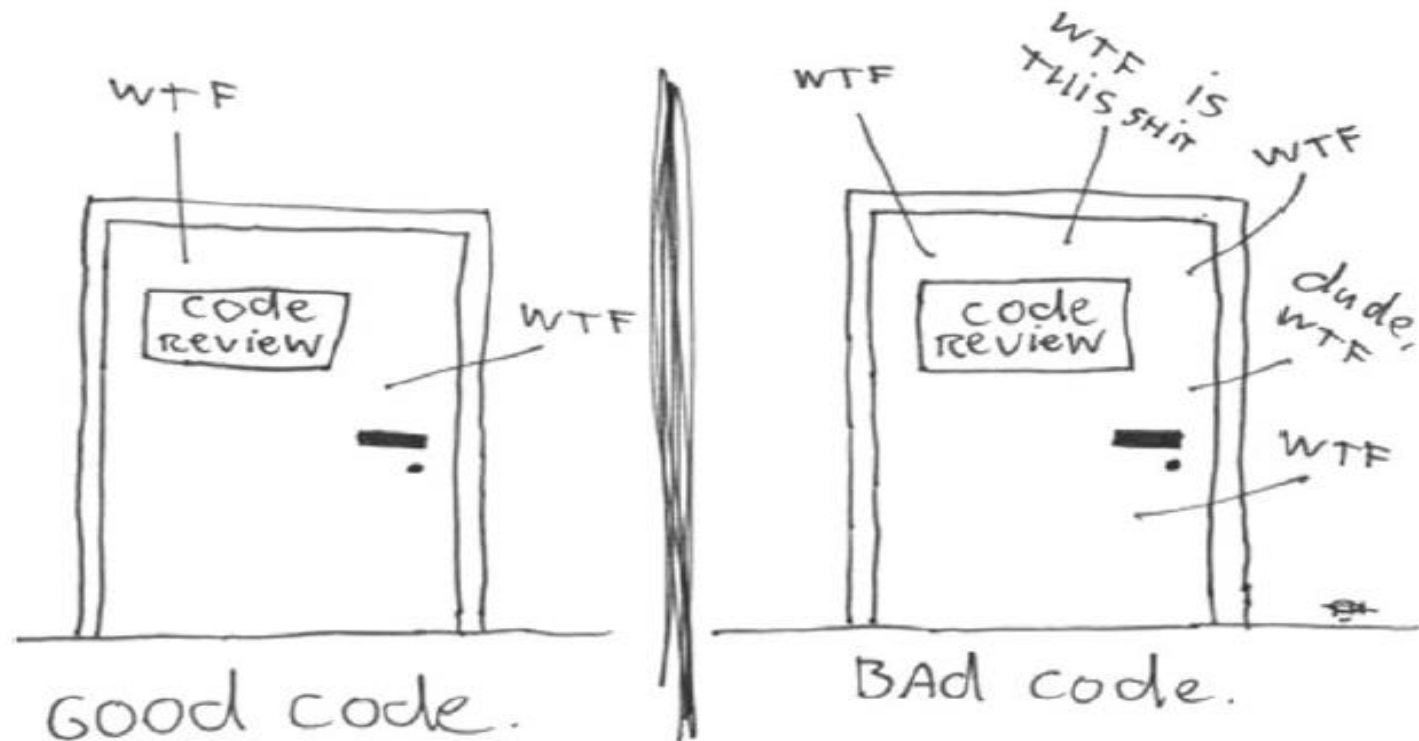
David Mytton, founder ServerDensity,

<http://www.sitepoint.com/coding-standards/>

Video: <http://www.youtube.com/watch?v=HjppjJ-xuvQ> (example of bad coding practice)

Intermezzo:
Sebaik apakah
kode sumber
anda?

The ONLY VALID MEASUREMENT
OF CODE QUALITY: WTFs/MINUTE



Reproduced with the kind permission of Thom Holwerda.
http://www.osnews.com/story/19266/WTFs_m

(c) 2008 Focus Shift

Intermezzo: Kode anda harus 'BERSIH'



Sumber:
Robert C. Martin, Clean Code, Prentice Hall

Kode yang 'BERSIH'

Kode yang:

1. Elegan & Efisien
2. Simple & Direct
3. Mempunyai *meaningful names*
4. Mempunyai *minimal dependencies*
5. Lulus pada semua tes
6. Tidak ada duplikasi
7. Mempunyai jumlah *class, function, method*, dll sesedikit mungkin (*tiny abstraction*)
8. Mudah dibaca oleh orang lain

The Need of Good Coding Practices (1)

- Readability
 - Program harus dapat dibaca dan dipahami dengan cepat dan baik oleh diri sendiri maupun orang lain
- Maintainability
 - Program dengan readability yang baik akan lebih mudah dipelihara
- Menghindari **code bad smells**
 - Tidak ada hubungannya dengan eksekusi, tetapi lebih pada kebaikan source code

The Need of Good Coding Practices (2)

- Dibutuhkan standar yang sama dalam penulisan kode
 - *Practice* yang umum dilakukan, setiap perusahaan bisa menetapkan *coding standard* sendiri
- Beberapa *practice* bila dilakukan dengan baik bahkan bisa mengarahkan pada program yang benar

Konvensi Penting IF1210

- Standar blok program
- Indentasi
- Pemakaian komentar dengan wajar dan baik
- Nama-nama yang baik dan *meaningful* untuk type, variabel, konstanta, fungsi/prosedur, dll.
- Hanya menulis kode yang memang dipakai
- Pemakaian *construct* dasar program dengan sebaik-baiknya
- Penggunaan *standard schema*

Standar Blok Program

- Blok program prosedural terdiri atas:
 - Header program: judul program dan spesifikasi
 - Kamus
 - Algoritma
- **Penting**: Jika Anda menulis program utuh (misalnya di praktikum) selalu tuliskan:
 - Identitas pemrogram (NIM>Nama)
 - Tanggal penulisan program → menunjukkan versi

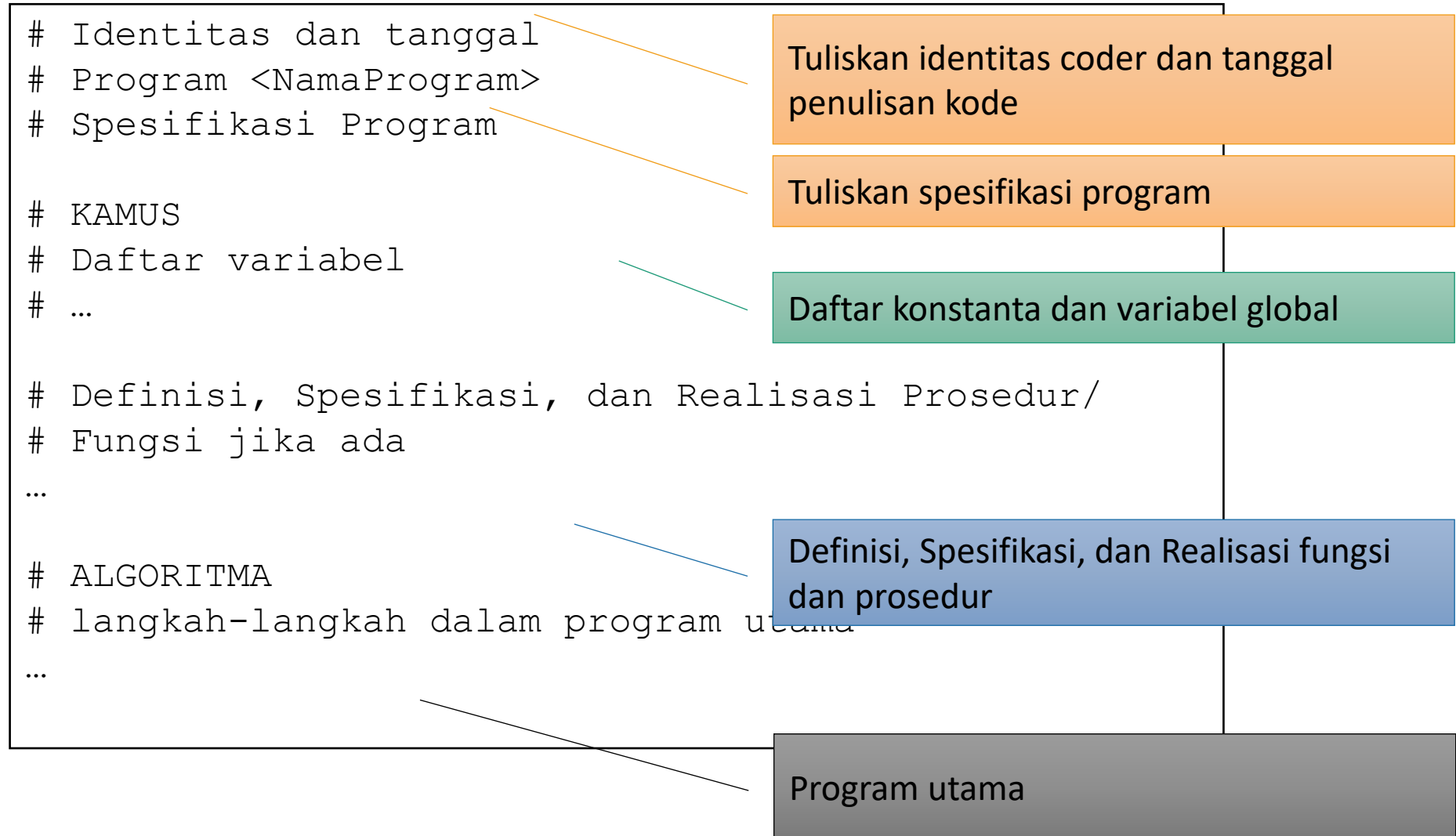
Standar Blok Program dalam Notasi Algoritmik

Program <JudulProgram>
{ Spesifikasi Program }

KAMUS
{ Deklarasi type, variabel, konstanta, fungsi, prosedur }

ALGORITMA
{ Deretan langkah algoritmik untuk penyelesaian persoalan }
{ **Memanfaatkan notasi algoritmik** }

Standar Blok Program Python



Indentasi (1)

- Gunakan indentasi yang semakin menjorok ke dalam untuk menandai setiap *inner block*
- Khusus untuk program yang diketik:
 - Gunakan spasi dengan jumlah yang sama untuk setiap indentasi baru
 - Tab dan spasi tidak bisa dicampur dengan baik → Gunakan tab dan spasi dengan konsisten

Indentasi (2)

- **Indentasi sangat penting di notasi algoritmik dan Python** karena menandai blok program
 - Contoh: 2 potongan kode ini berbeda eksekusinya

```
...  
x ← -1  
if (x ≥ 0) then  
    output("Mungkin positif")  
    output("Mungkin nol")  
...
```

```
...  
x ← -1  
if (x ≥ 0) then  
    output("Mungkin positif")  
output("Mungkin nol")  
...
```


Indentasi (3)

Blok program di Python

- Blok program sangat penting di Python.
- Jika instruksi berada dalam 1 blok, maka indentasi harus rapi. Jika tidak, akan *error*.

```
a = int(input("Masukkan angka = "))  
if (a > 50):  
    print ("Hello World!")  
    print ("bye")  
else: # a <= 50  
    print ("Hello Darling!")  
    print ("bye bye")
```



OK!

```
a = int(input("Masukkan angka = "))  
if (a > 50):  
    print ("Hello World!")  
    print ("bye")  
else: # a <= 50  
    print ("Hello Darling!")  
    print ("bye bye")
```



Error!

Komentar

- Komentar wajib dalam program (konvensi):
 - identitas + tanggal, spesifikasi program
 - Spesifikasi fungsi dan prosedur
- Komentar untuk hal-hal yang penting
 - Tidak berlebihan sehingga teks penuh komentar

Nama-nama yang baik dan *meaningful* (1)

- Gunakan 1 nama untuk **1 buah keperluan** saja
 - Hindari menggunakan variabel/subprogram dengan nama sama untuk keperluan yang berbeda
- Gunakan nama yang memang menggambarkan hal yang direpresentasikan (*meaningful*)
 - hindari menggunakan nama yang membingungkan atau tidak merepresentasikan apa pun

```
procedure Analisis ( ... )  
nnmsspd : integer  
akucintakamu : character
```

Nama-nama yang baik dan *meaningful* (2) - Variabel

- Variabel:
 - Perhatikan nama-nama yang memiliki arti dan penggunaan yang dikenal umum: e.g. flag, found, count, sum, idx, min, max, ... → hindari menggunakannya untuk keperluan lain
 - i, j, k hanya digunakan dalam *control loop*

Hanya menulis kode yang memang dipakai

- Bagian kode yang berlebihan tidak akan menimbulkan compile-error
 - Beberapa compiler memberikan warning
- Hindari menuliskan bagian kode yang tidak dipakai sama sekali
 - Khususnya dalam menuliskan program utuh (di praktikum dan tugas besar)

Pemakaian *construct* dasar program dengan sebaik-baiknya

- Perhatikan kembali *construct* dasar program prosedural dan gunakan sesuai dengan tujuannya dengan sebaik-baiknya
- Contoh:
 - Analisis kasus:
 - harus disjoint dan complete (walaupun mungkin tidak semua kasus dituliskan aksi yang eksplisit)
 - Kapan menggunakan if-then-else, depend-on, dll.
 - Loop: gunakan jenis loop yang benar → akan diulas kembali dalam skema pengulangan

Penggunaan *standard schema*

- Menggunakan skema 'standar' yang disepakati
- Dalam kuliah IF1210 akan dibahas lebih lanjut beberapa skema dasar dan standar:
 - Skema proses validasi
 - Skema pengulangan/pemrosesan sekuensial
 - Skema pemrosesan array
 - Skema pemrosesan file I/O

SELAMAT BELAJAR