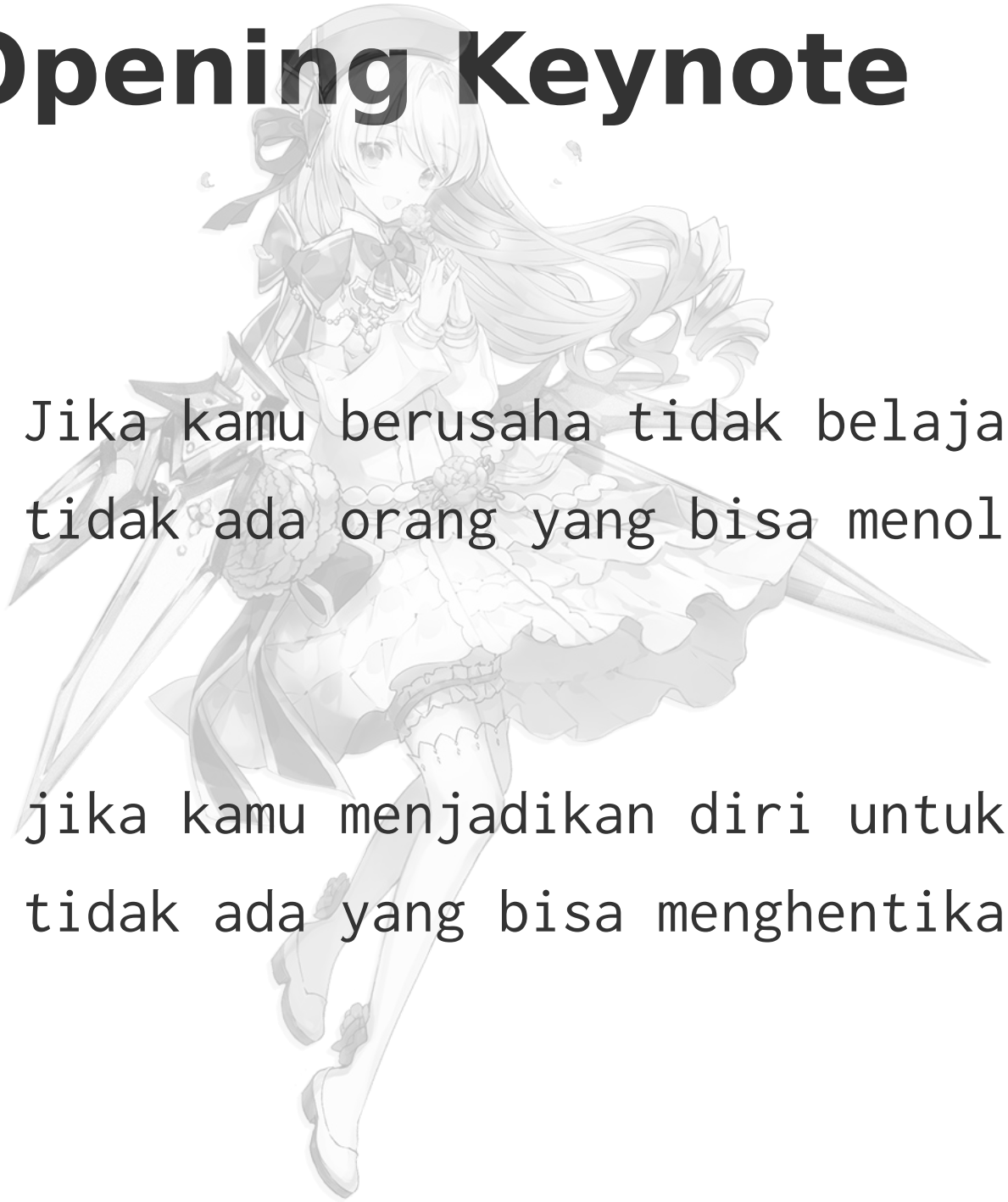
Algoritma dan Pemrograman

Struktur Dasar Program dan Diagram Alur

Opening Keynote



Jika kamu berusaha tidak belajar,
tidak ada orang yang bisa menolongmu

jika kamu menjadikan diri untuk belajar,
tidak ada yang bisa menghentikan mu.

Subtopik

- Input-Process-Output pada Program
- Mendesain Flowchart dari suatu algoritma
- Deklarasi dan Pengunan variable , konstanta ,ekpresi dan tipe data
- Input/Output
- Sekuens
- Contoh Kasus

Capaian

- Anda mampu membuat flowchart dari suatu algoritma penyelesaian masalah tertentu
- Anda mampu memahami pseudo-code dan mengaplikasikanya
- Anda mampu memahami dan mengunakan variable , expresi , input dan output.

Syarat Material

Untuk mengikuti rangkaian materi pada slide ini ada prasyarat yang perlu dipenuhi :

- Interpreter Ruby pada media yang akan anda gunakan baik komputer , Handphone atau mengunakan situs daring.
- Saran, gunakan FOSS (Free – Open Source Software) / Perangkat Lunak Bebas Gratis.

Syarat Mental

- Persiapkan mental anda , jadi pelajar yang proaktif bukan pengemis yang reaktif
- Gunakan Akal dan Daya Kritis anda
- Berasa Ingin tahu dan eksplorasi
- Hadapi masalah , pecahkan serta berani mengotori tangan sendiri
- Jadila penanya yang cerdas , karena belajar dan pahami terlebih dahulu yang anda ingin tanyakan suatu kebermanfaatan.

Bacaan Lanjutan

- Berfikir Komputasional
- Pseudo-Code dan UML
- Clean Code : DRY , KISS , SOLID , dan lainya
- Struktur Data
- Pemrograman Berorientasi Objek

Pemrograman

Ruby

- Ruby merupakan bahasa pemrograman General-Purpose dan berparadigma Objek-Oriented.
- Diciptakan pada tahun 1995 oleh Yukihiro “Matz” Matsumoto.
- Pada seri pembelajaran ini secara esklusif mengunakan Ruby.
- Untuk pengunaan dasar kita akan mengunakan paradigma Prosedural.
- Case Sensitive , besar-kecil huruf berpengaruh

Contoh Kerja Program

Input	Proses	Output
Input A Input B	A = A+ B B = A + B	Output A Output B

Ruby

A = gets.to_i B = gets.to_i	A = A + B B = A + B	puts A puts b

```
 1 # Kamus
 2 a = 3 # gets.to_i
 3 b = 4 # gets.to_i
 4
 5 # Algoritma
 6 a = a + b # 7
 7 b = a + b # ??
 8
 9 puts a # ??
10 puts b # ??
11
```

Kamus

Algoritma

```
 1 # Kamus
 2 a = 3 # gets.to_i
 3 b = 4 # gets.to_i
 4
 5 # Algoritma
 6 a = a + b # 7
 7 b = a + b # ??
 8
 9 puts a # ??
10 puts b # ??
11
```

Kamus

- Kamus dipakai untuk mendeklarasikan / menyatakan nama yang digunakan pada program.
- Deklarasi bukan Instruksi
- Definisi yang diterima :
  - Variable
  - Konstanta

Jenis Tipe Data

Setiap data memiliki komponen penyusun sendiri

- Data Ukuran Sandal berbeda dengan Data Nama Seseorang
  - Data Ukuran Sendal : 38 (terdiri atas angka)
  - Data Nama : Elodia Kartini (terdiri dari karakter alphabet)

Jenis Tipe Data

Tipe data primtif / dasar

- Integer
- Float
- String
- Boolean

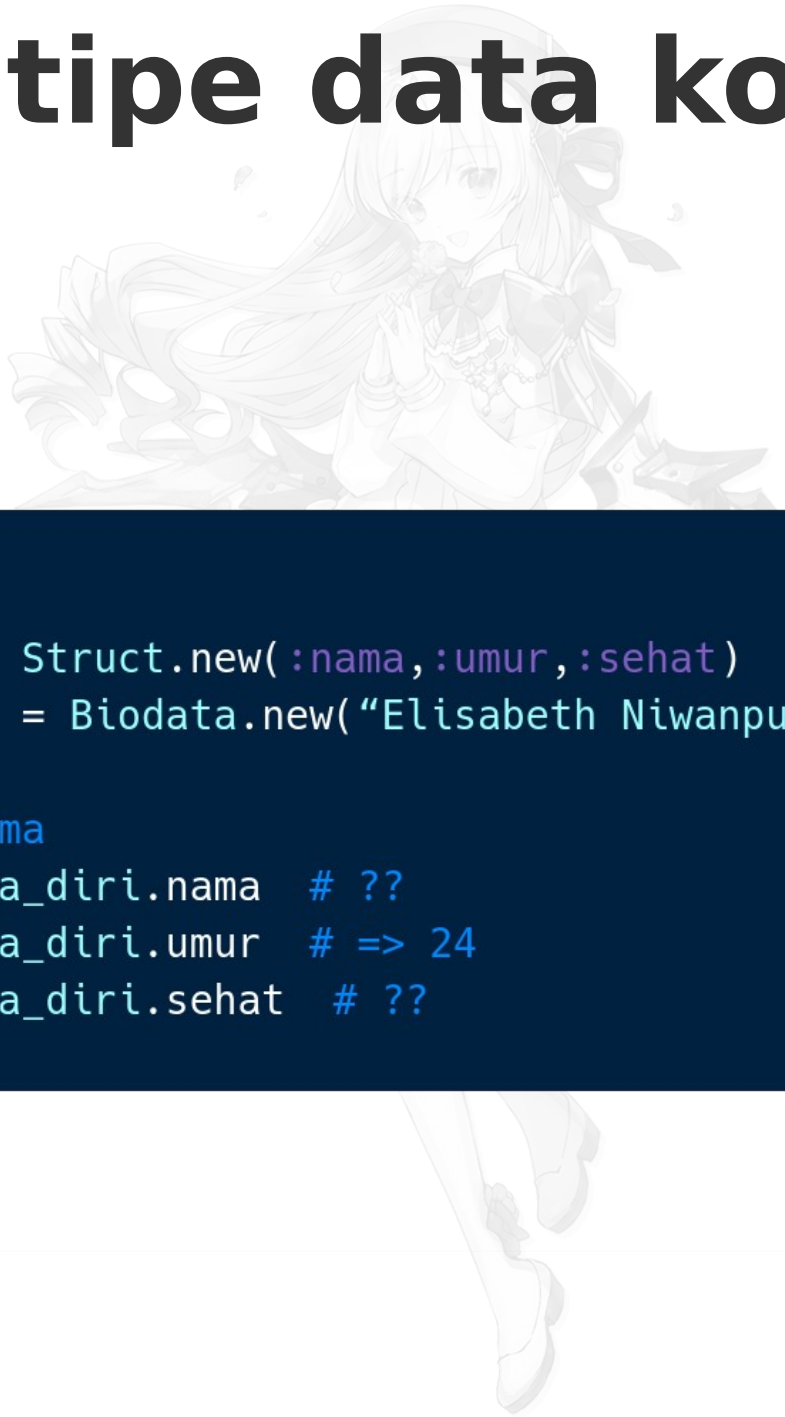
Tipe data artifisial / bentukan dan turunan

- Tipe data Biodata
  - Nama : String
  - Umur : Integer
  - Sehat : Boolean

Contoh Tipe Data

- Nama → String , contoh : “Kartini”,”Niwanputri”
- Tanggal → String , contoh : “21-04-1999”
- Tinggi Badan → Integer / Float , contoh : 165, 165.23
- Berat Badan → Integer / Float , contoh : 42 , 42.23
- Sehat → Boolean, contoh : True / False

Contoh tipe data komposit



```ruby
#Kamus
Biodata = Struct.new(:nama,:umur,:sehat)
data_diri = Biodata.new(“Elisabeth Niwanputri”,24,true)

# Algoritma
puts  data_diri.nama  # ??
puts  data_diri.umur  # => 24
puts  data_diri.sehat  # ??
```

Variable

- Variable merupakan tempat menyimpan data dengan ber-tipe data sesuai dengan deklarasi.
- Pada Ruby Deklasi variable harus disertai nilai , karena tipe data Ruby besifat Loosly Type.

Contoh:

```
umur = 24
```

- Variable umur di isi dengan nilai 24
- 24 nilai bertipe integer , maka tipe data variable umur adalah integer

```
nama = “Vina”
```

- Variable nama disi dengan nilai “Vina”
- “Vina” bernilai String maka. Demikian juga tipe data variable nama.

Loosly Type

- Pada Ruby tipe data variable ditentukan dengan tipe data nilai yang diberikan (*assigning*). Deklarasi variable pada Ruby dapat dikatakan Pemberian nilai ketimbang deklarasi variable.
- Untuk mengetahui tipe data variable yang sudah terdeklrasi dengan data mengunakan , .class pada nama variable.
  - `variable_nama = “Elisabeth Niwanputri”`
  - `variable_nama.class #=> String`

Operasi pada Tipe Data

- Operasi perhitungan seperti +,-,*,/ merupakan operasi untuk melakukan hitungan dengan angka.
- Tipe data berbeda bukan berarti memiliki arti yang beda
  - “ Ru “ + “ by “ → “Ruby”
- Namun tidak semua operasi dapat digunakan untuk semua tipe data
  - “Aku mandi 0 ” * “Sehari” → ERROR

Operasi tipe data primitif

Daftar ini namun ini tidak menutup kemungkinan dari arti operasi itu sendiri.

Interger

- * , / , + , - ,% , < , > , <= , >= ,== ,!=

String

- == , !=

Boolean

- && , || ,!

Penamaan Variable

- Nama diawali dengan huruf kemudian setelah di ikuti angka / huruf
  - Nama tidak mengandung tanda baca dan spasi
  - Gunakan *underscore*(_) sebagai pemisah.
- Nama variable mudah dimengerti dan mengambarkan tujuan data didalamnya.
- Case-Sensitif berpengaruh

Penamaan Variable

- `13e5ok_L1buR = true`
  - Penamaan seperti ini tidak bermakna serta salah

- `nama_lagu = “Lilac for Anabel — Apo11o program“`
  - Ini mengambarkan variable berisi data tentang nama lagu

- Apakah penamaan variable dibawah benar ?
  - `Kucing`
  - `_ikan`
  - `3`
  - `waktu_siang`

Konstanta

- Konstanta Serupa dengan Variable , namun diawali dengan Huruf Besar atau semua Huruf Besar semua.
- Sekali Konstanta diberi nilai setelah nya tidak bisa diubah.

- Contoh

```
GOLDEN_RASIO = 1.615
Tuhan = “Tidak ada”
Matahari_Panas = true
```

Algoritma

Algoritma

Dalam Konteks Umum

- Merupakan suatu langka-langkah sistematis yang digunakan untuk menyelesaikan masalah.

Dalam Pemrograman

- Merupakan alur instruksi dalam teks algoritmis program yang terurut untuk menyelesaikan masalah.

Teks Algoritmis

Yaitu ,

- Perintah dasar (output/input,assignment)
- Perintah berurut (Sekuensial)
- Analisis Kasus (jika maka , kenapa)
- Pengulangan

Kriteria Algoritma yang baik

- Memiliki Input
  - Algoritma memiliki nilai masuk
- Memiliki Output
  - Algoritma memiliki nilai keluar
- Memiliki Batasan
- Memiliki Kepastian
- Effisien

Perintah dasar

- Pemberian Nilai
- Perbandingan
  - Persamaan (==)
  - Pertidaksaaman (!=)
- Operasi relasional
  - >= , <= , < , >
- Operasi aritmatika
  - * ,/ , + ,-

Input

- Input pemberian nilai terdapat dua cara
  - Assignment
    - Contoh:
      - tanggal = “20-04-2012”
  - Pirantin Inputan
    - Contoh:
      - nama = gets.to_s
- Untuk menampilkan kelayar dapat dengan `puts` atau `print`.

Pemberian nilai (Asignment)

- Pada Ruby deklarasi variable bersifat pemberian nilai (Loosely Type) , jadi tidak ada variable tanpa nilai.
- Tanpa Loosely type , kita bisa mendeklarasi variable , tanpa inisialisasi nilai.
- Ruas Kiri = Ruas Kanan
  - Ruas Kiri
    - Harus Variable
  - Ruas Kanan
    - Harus expressi
    - Contoh
      - “luas segitiga”
      - luas = alas * tinggi

Expresi

- Expresi Aritmatika
  - 2 * jari_jari_kotak
  - A + B
- Expresi Relasional
  - A > B
  - X != Y
- Expresi Logika
  - A && B
  - C | | B

Komentar

- Dalam Bahasa Pemrograman , Komentar merupakan bagian yang tidak dieksekusi isi nya
  - Bagian ini ditujukan untuk meberikan informasi tentang keterangan , catatan penting , dan pengingat.
- Dalam Ruby , Komentar dituliskan dengan
  - # apa awal baris , kalimat komentar

    Contoh
    - A = C + D # A untuk Hasil , C dan D dapat dari input user.

Aksi Sekuensial

- Aksi sekuensial
  - Barisan intruksi / aksi yang dieksekusi komputer sesuai dengan urutan penulisanya
- Setiap aksi memiliki dampak ke program
  - Setiap aksi haru definitif (memiliki awal dan akhir yang jelas)
  - Instruksi ditulis sesuai urutan

Contoh Aksi Sekuensial

```
1 puts "Masukan Nama"
2 nama = gets.to_s
3
4 puts "Analisis"
5 print "Panjang String: "
6 print nama.size
```

Contoh Algoritma

- Persoalan
  - Bagaimana menghitung perkalian dengan menambahkanya?
- Definisikan Masalah
  - Input : angka penambah dan pengali
  - Output : hasil tambahan
- Langkah penyelesaian
  - 1 . baca angka penambah
  - 2 . baca angka pengali
  - 3 . Tambahkan angka penambah berulang sebanyak angka pengali

Pseudo-code

- Pseudo-code merupakan struktur kode lojik yang dengan bahasa lebih manusiawi.

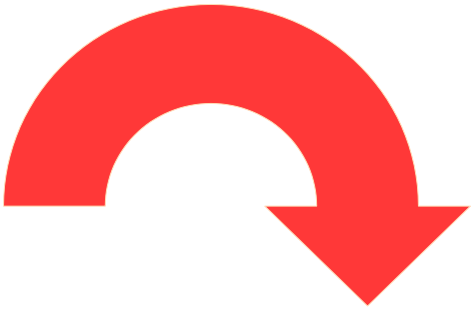
- Tidak ada standar terlalu khusus, biasanya dalam bahasa yang mudah dipahami atau notasi matematis

Contoh Pseudo Code

```
1 ambil input masukan ke angka pengali
2 ambil input masukan ke var angka penambah
3 pasang nilai 0 ke var hasil
4
5 hasil tambahkan dengan angka penambah
6 ditambah dengan dirinya
7 sebanyak nilai angka pengali
8
9 tampilkan ke layar hasil
```

Contoh Pseudo code

```
1 ambil input masukan ke angka pengali
2 ambil input masukan ke var angka penambah
3 pasang nilai 0 ke var hasil
4
5 hasil tambahkan dengan angka penambah
6 ditambah dengan dirinya
7 sebanyak nilai angka pengali
8
9 tampilkan ke layar hasil
```



Penerjemaahan (perubahan)
dari Pseudo-Code ke Ruby code

```ruby
1 angka_pengali = gets.to_i
2 angka_penambah = gets.to_i
3 hasil = 0
4
5 angka_pengali.times{ |pengali| hasil += angka_penambah }
6
7 puts hasil
```

Contoh Kasus

- Permasalahan
  - Buatlah penghitung volume bola
- Definisi Masalah
  - Input : Jari – Jari Bola
  - Output : Volume Bola

Contoh Kasus (Pseudo)

```
1 ambil input dari user masukan ke variable jari_jari.
2
3 masukin ke rumus volum bola (4/3) * 3.1415 * jari_jari^3
4 dan tampung dalam variable jari_jari.
5
6 tampilkan nilai variable jari_jari.
```

Contoh Kasus (Ruby)

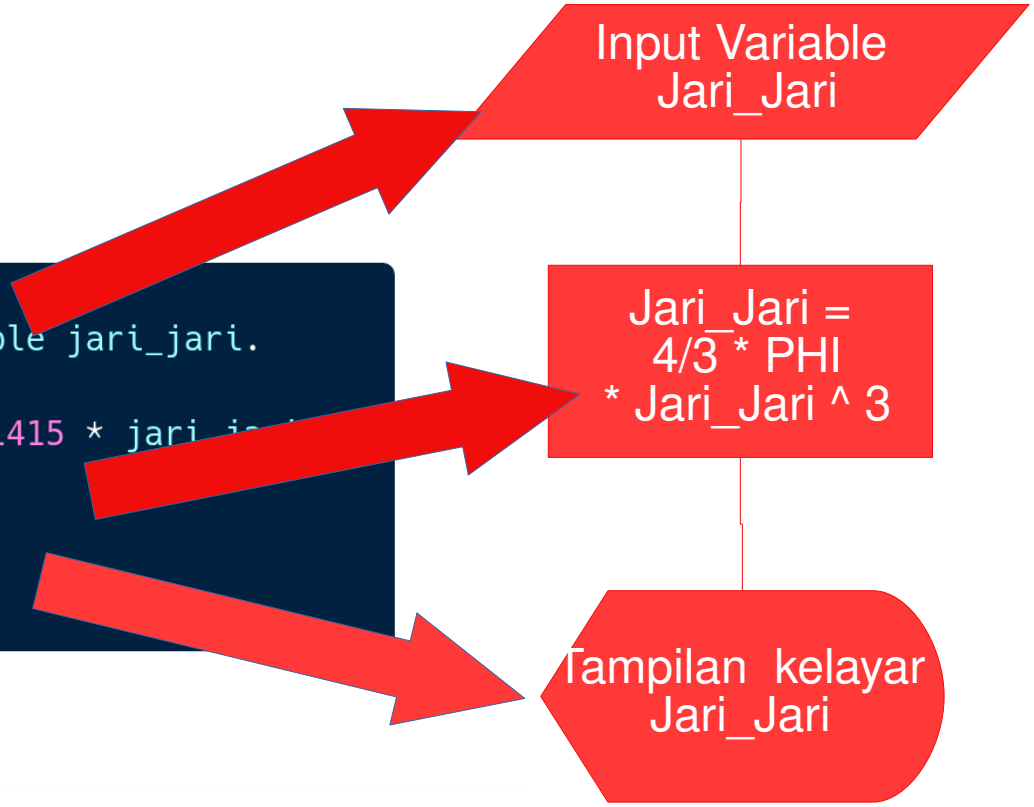
```ruby
# puts agar lebih informatif saat digunakan
puts " Masukan Jari - Jari ":
jari_jari = gets.to_f

PHI = 3.1415
jari_ jari = (4.0/3.0) * PHI * (jari_jari**3)

puts " Volume Bola : "
puts jari_jari
```

Hasil keluaran program :

```
root ~  ruby volume_bola.rb
Masukan Jari Jari Bola
3
Volume Bola :
113.094
root ~
```

Flowchart (Diagram Alur)

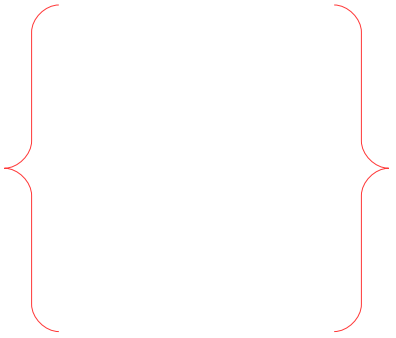
```
1 ambil input dari user masukan ke variable jari_jari.
2
3 masukin ke rumus volum bola (4/3) * 3.1415 * jari_ja
4 dan tampung dalam variable jari_jari.
5
6 tampilkan nilai variable jari_jari.
```



Kasus Algoritma

- Permasalahan
  - Buatlah program penghitung dari detik ke jam dan menit.
- Definisi masalah
  - Input :
    - Detik (Integer)
  - Output
    - Menit (Integer)
    - Jam (Integer)

- Latihan
  - Bentuk dalam Pseudo-code
  - Bentuk dalam bahasa Ruby



Memulai Ruby dengan Script

- Buka text editor kalian bebas.
- Contoh ini pakai Vim. Dan simpan pada direktori tertentu dengan akhiran .rb

```
1 #!/usr/bin/ruby
2
3 puts "Masukan Nama"
4 nama = gets.to_s
5
6 puts "Analisis: "
7 puts "Panjang : #{nama.size}"
~
```

Memulai Ruby dengan Script

- Jalan pada terminal , “ ruby <alamat/nama script kalian>.rb “

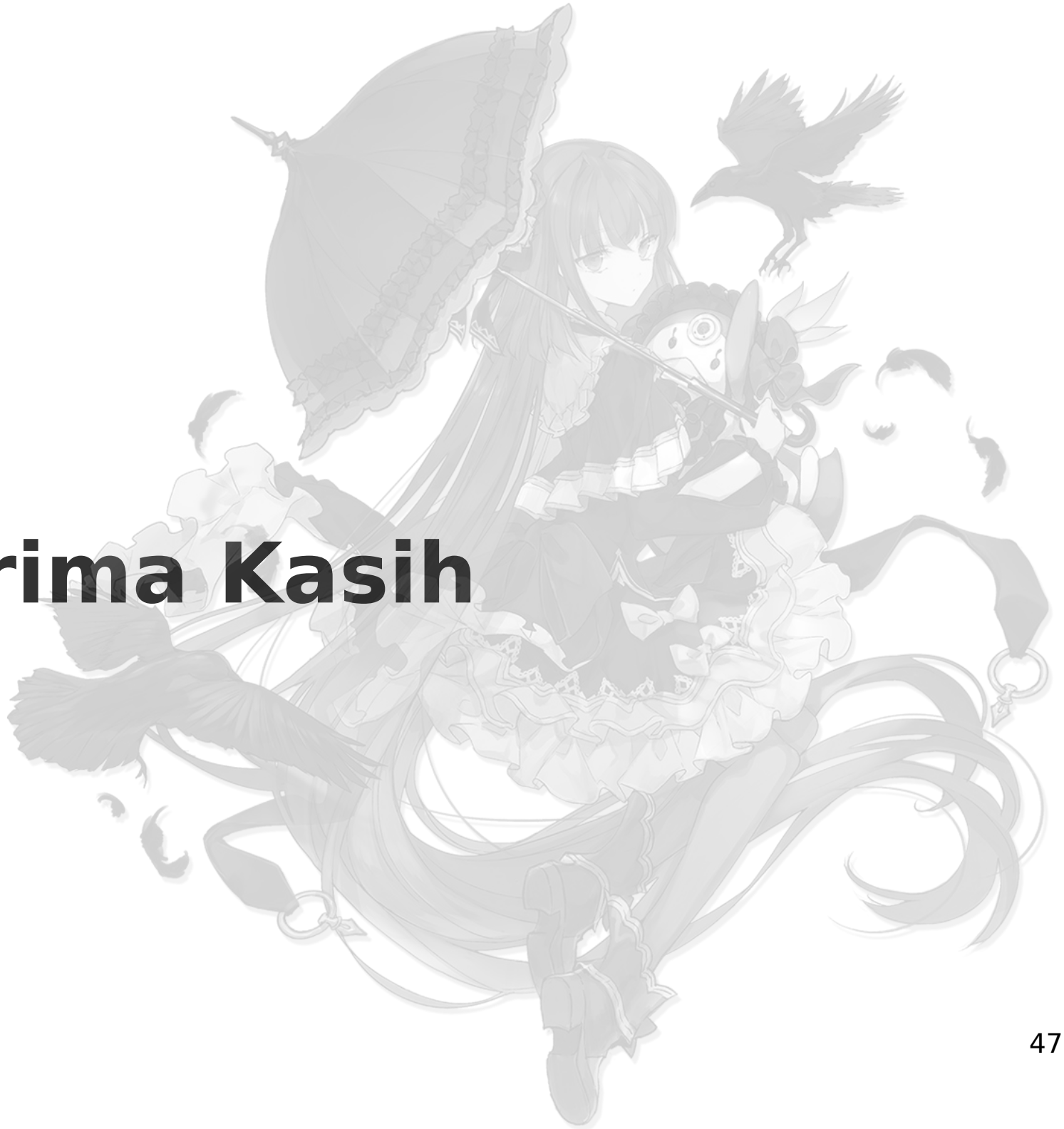
```
root ~  vim hallo.rb
root ~  ruby hallo.rb
Masukan Nama
Elisabeth Kartini
Analisis:
Panjang : 18
root ~
```

Closing Keynote

Pengetahuan itu tak terbatas.

– ConneR

> “Kita tidak bisa mengatur angin , tapi kita bisa menaru jangkar “ – Irish Proverb



Terima Kasih

Dalam nama perjuangan

- Materi ini dibuat sebagai bentuk perlawanan terhadap komersialisasi pendidikan dan pengendalian informasi
- Menjunjung kebebasan informasi dan pencerdasan umum
- Hak cipta bebas merdeka , setiap orang dianjurkan dan dinasehatkan untuk mengopi ,mencetak , mengganda, menyebar isi serta materi – materi didalamnya.