# Algoritma dan Pemrograman

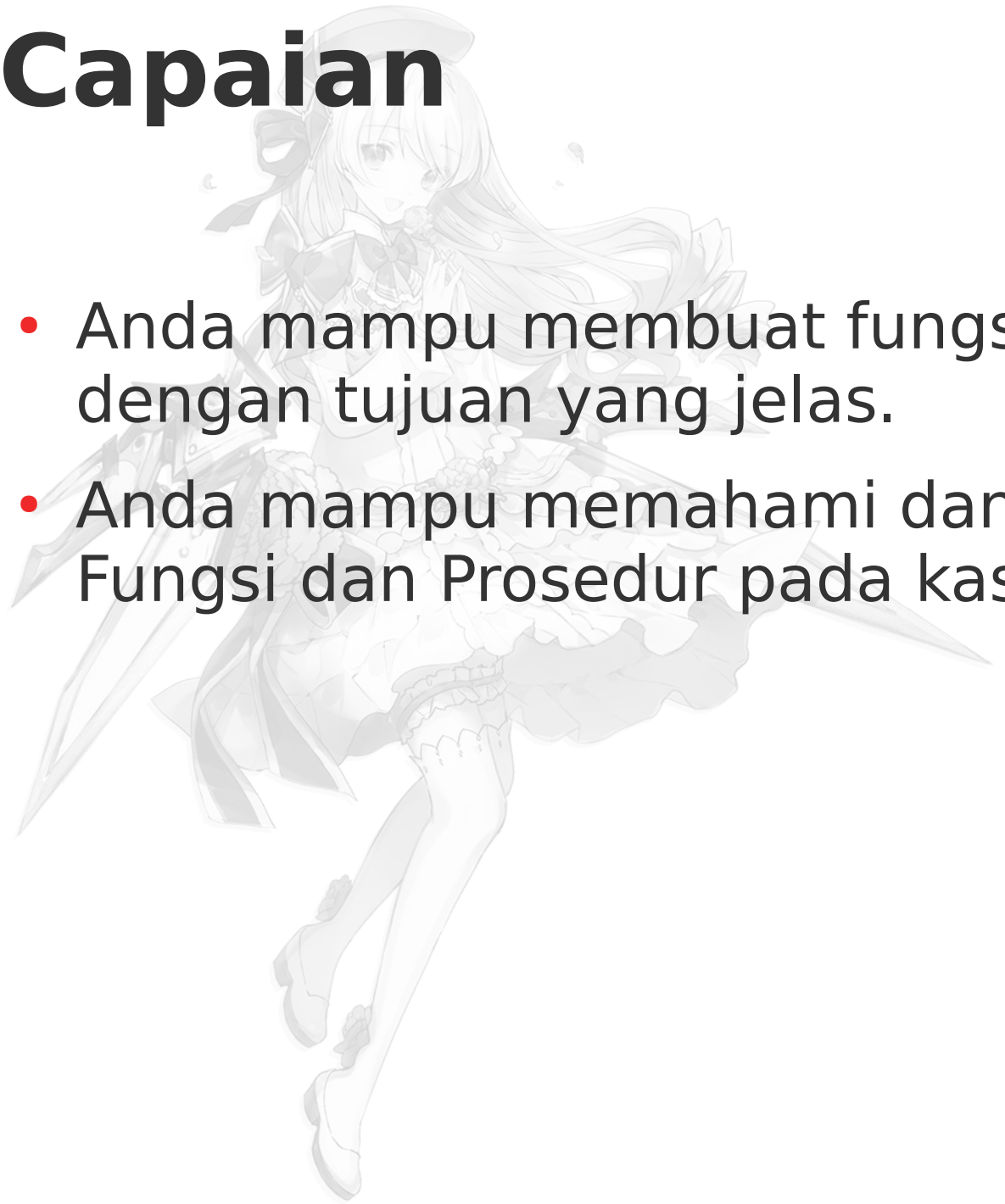
## Subprogram

# Opening Keynote

# Subtopik

- Pengenalan Prosedur
- Pengenalan Fungsi
- Rekursif

# Capaian



- Anda mampu membuat fungsi dan prosedur dengan tujuan yang jelas.
- Anda mampu memahami dan mengunakan Fungsi dan Prosedur pada kasus tertentu.

# 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

# Algoritma

# Pengenal Subprogram

- Merupakan kumpulan set intruksi/blok kode yang biasa digunakan dalam operasi program.
- Contoh Subprogram
  - Fungsi
  - Prosedur

# Pengenalan Fungsi

- Fungsi semacam formula yang menerima argumen untuk mengoperasinya dengan menghasilkan nilai kembalian ( *return* )
- Kamus dan parameter dalam subprogram hanya bisa diakses oleh subprogram itu sendiri. Ini disebut lingkup variable ( *scope* )
- Global variable adalah variable yang dapat diakses oleh semua subprogram.

# Langkah membuat fungsi

- Mendefinisikan Fungsi
  - Tentukan nama fungsi,bila yang menunjukan kegunanya akan lebih baik.
  - Mendefinisikan parameter /argumen .
  - Menentukan nilai kembalian
- Menrelisasikan Fungsi
  - membuat algoritma fungsi :

    input parameter → hasil nilai kembalian
- Memanggil Fungsi
  - Memangil fungsi dengan parameter aktual

```
1 def nama_fungsi (parameter_fungsi,....)
2   # KAMUS
3   #     kumpulan variable dan data yang dibutuhkan
4   # ALGORITMA
5   #     langkah / langkah instruksi
6 end
```

**Contoh Fungsi .**

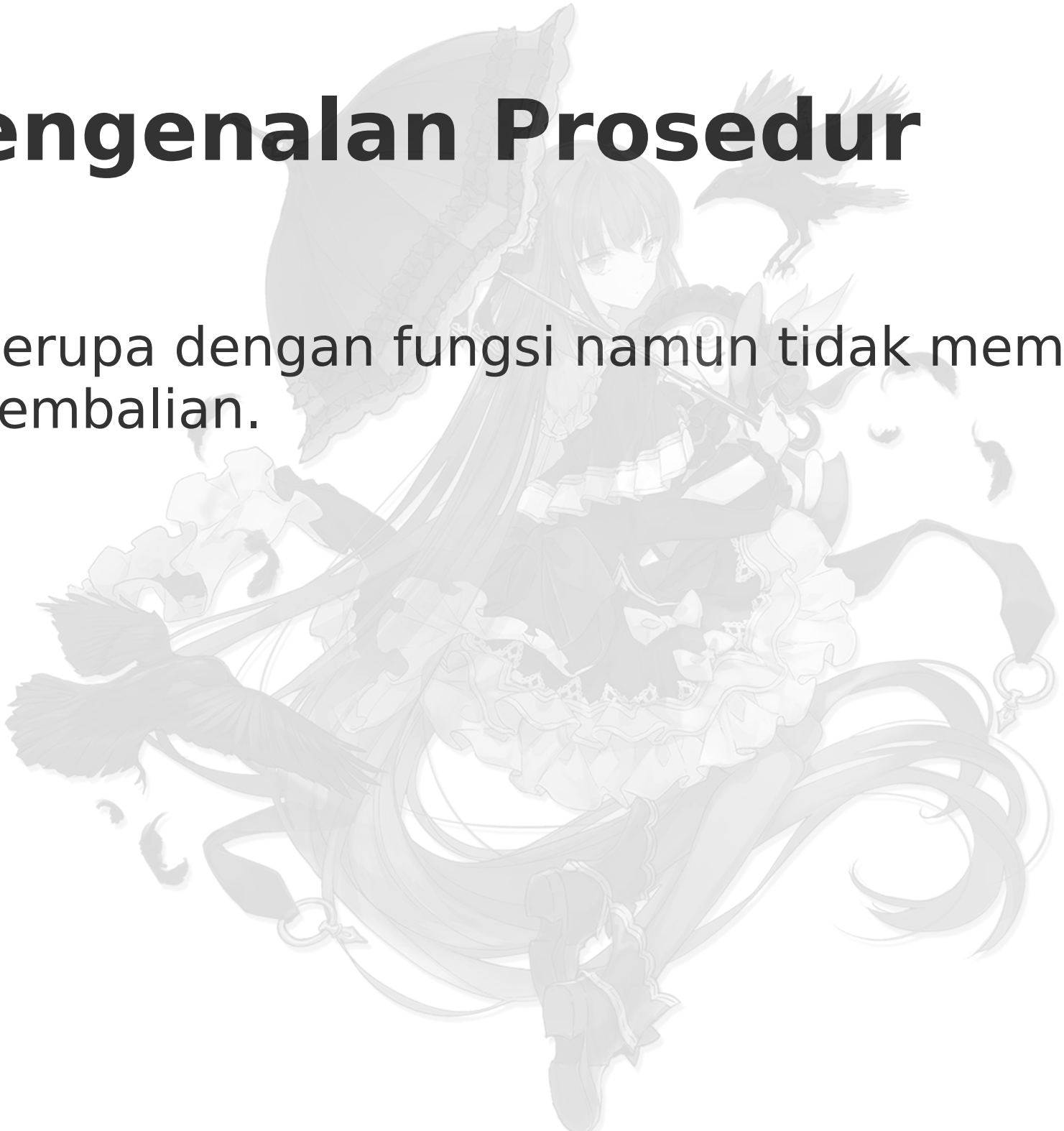
**Karena di subprgram memiliki Kamus dan Algoritma nya sendiri layaknya program namun dalam program.**

```ruby
Biodata = Struct.new(:nama,:tanggal_lahir,:filsafat)

def biodata_menjadi_hash biodata_param
  # KAMUS
  hash_biodata = {}

  # ALGORITMA
  hash_biodata["nama"] = biodata_param.nama
  hash_biodata["tanggal_lahir"] = biodata_param.tanggal_lahir
  hash_biodata["filsafat"] = biodata_param.filsafat
  return hash_biodata
end

aku = Biodata.new("Elisabeth Niwanputri",24,"Nihilisme")

biodata_menjadi_hash(aku)
# keluaran
#   {"nama" => "Elisabeth Niwanputri",
#   "tanggal_lahir" => 24,
#   "filsafat" => "Nihilisme"}
```

Deklarasi Fungsi biodata_menjadi_hash

Pemnaggilan Fungsi : biodata_menjadi_hash Paramater: aku

# Pengenalan Prosedur



- Serupa dengan fungsi namun tidak memiliki kembalian.

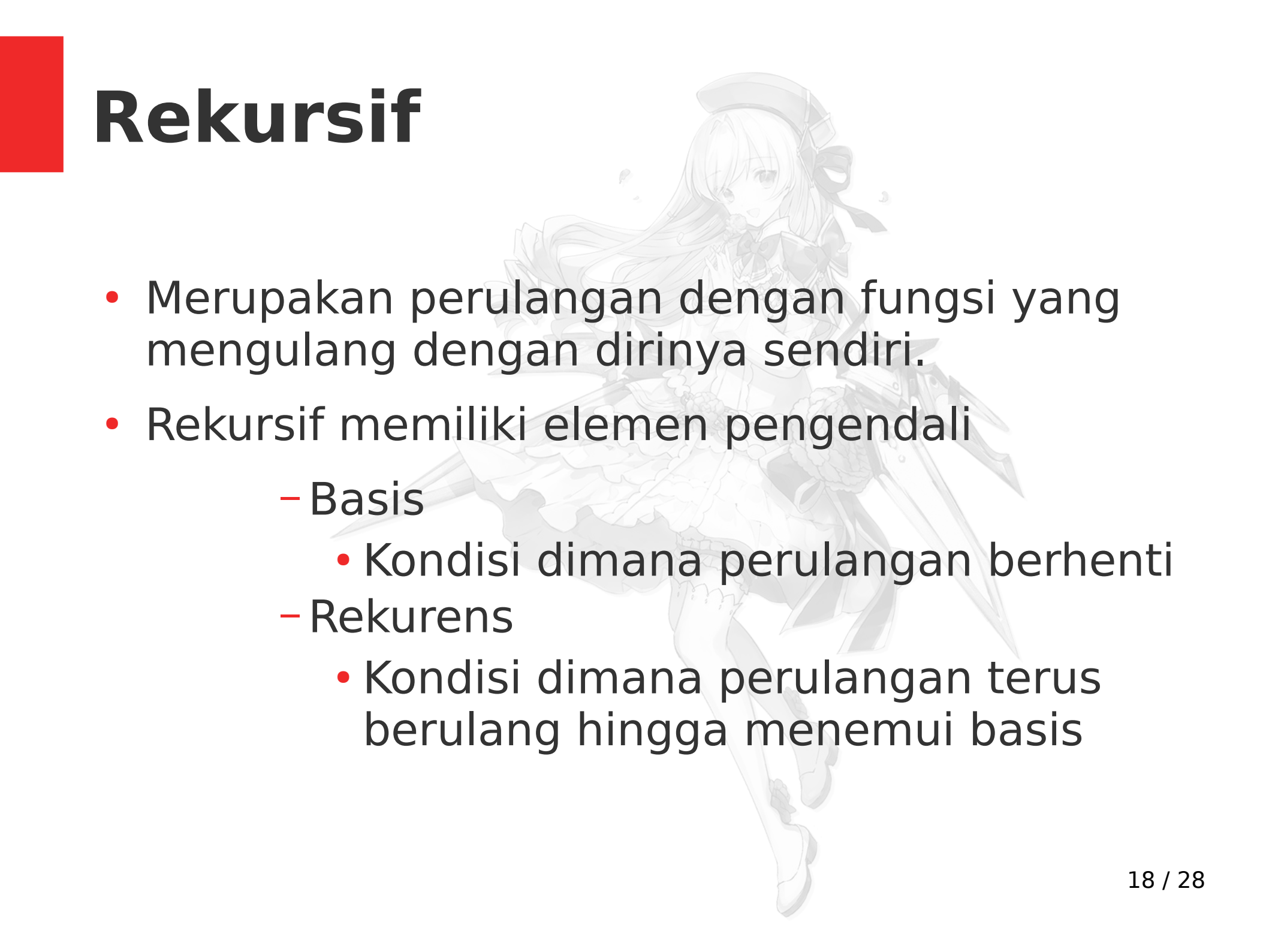
```ruby
Biodata = Strcut.new(:nama,:tanggal_lahir,:filsafat)

def cetak_biodata biodata_param
  # KAMUS
  # ALGORITMA
  puts "--- Biodata ----------"
  puts " Nama : "
  print biodata_param.nama
  puts " Tanggal Lahir : "
  print biodata_param.tanggal_lahir
  puts " Filsafat : "
  print biodata_param.filsafat
  puts "----------------------"
end

aku = Biodata.new("Elisabeth Niwanputri",24,"Nihilisme")

cetak_biodata(aku)
# keluaran
#   --- Biodata ------------
#   Elisabeth Niwanputri
#   24
#   Nihilisme
#   ----------------------
```

# Parameter

- Pada subprogram terdapat parameter , yang menjadi input bagi subprogram.
- Parameter memiliki berapa jenis
  - Biasa
    - Untuk meneruskan parameter nilai dari pemanggilan. Harus ada.
  - Opsional
    - Parameter ini diberi nilai  atau tidak saat pemanggilan
  - Default ( bawaan )
    - Parameter akan bernilai default jika tidak diberi nilai saat pemanggilan

# Pemrograman

# Rekursif



- Merupakan perulangan dengan fungsi yang mengulang dengan dirinya sendiri.
- Rekursif memiliki elemen pengendali
  - Basis
    - Kondisi dimana perulangan berhenti
  - Rekurens
    - Kondisi dimana perulangan terus berulang hingga menemui basis

# Bentuk Umum Rekursif

```
1 def rekursif parameter
2   if basis
3   else
4     #rekurens
5   end
6 end
```

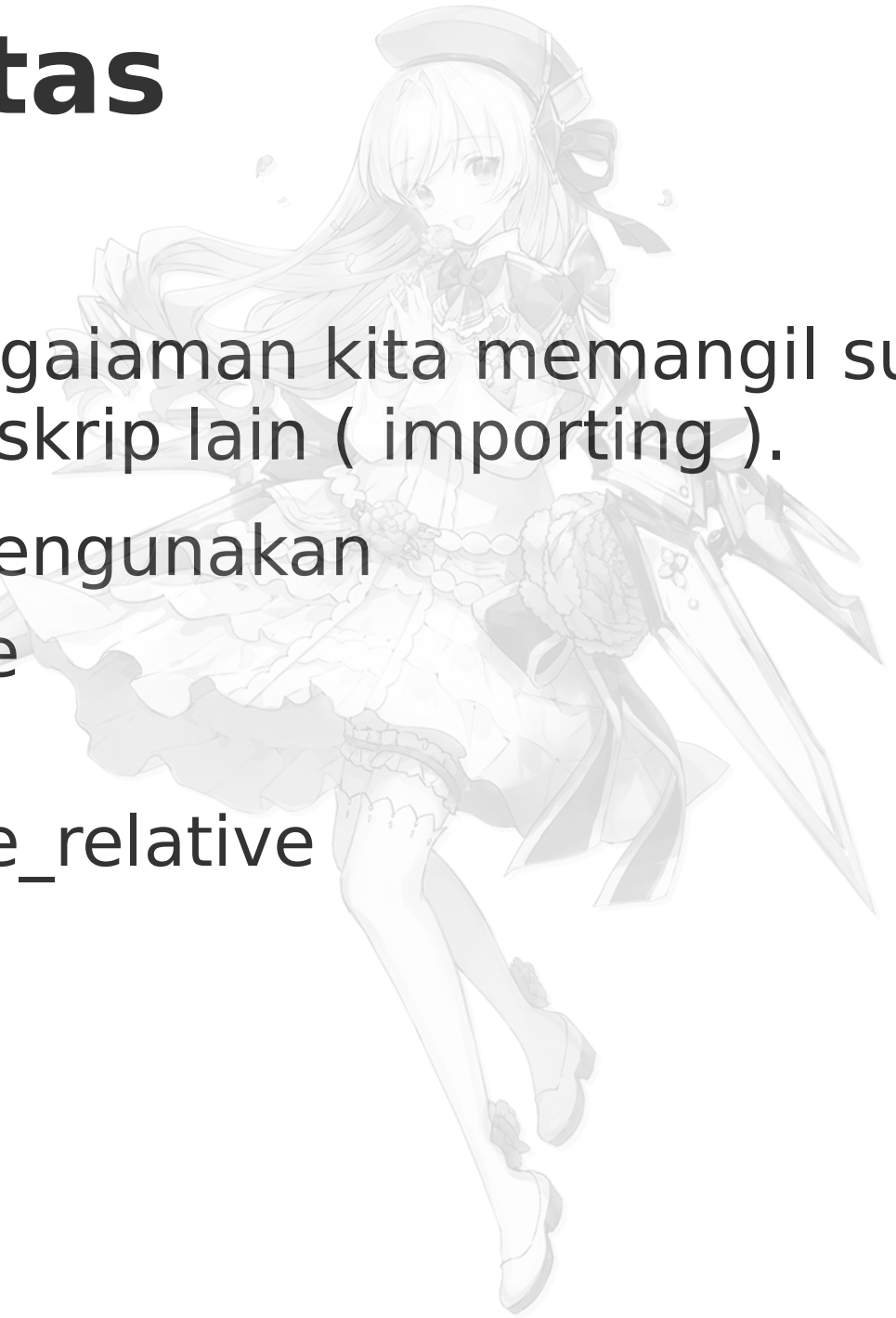
# Contoh Rekursif

```
1 def mencari_elmnt data_colleksi,index=0,target
2   if target == data_colleksi[index]
3     return index
4   else
5     return mencari_element(data_colleksi,index + 1,basis)
6   end
7 end
```

# Contoh Rekursif

```
1 album_lagu_klasik = ["VILA - Another Katharsis" ,"Xi vs ICE - Scherzo","Ice - Pandora's Box"]
2 def iterator_rekursif koleksi
3     if koleksi.empty?
4      return 0
5    else
6     puts koleksi.pop
7     return iterator_rekursif(koleksi)
8    end
9 end
10
11 iterator_rekursif(album_lagu_klasik)
```

# Modularitas



- Modularita bagaiaman kita memangil sub-program dari skrip lain ( importing ).
  - Dapat mengunakan
    - require
    - load
    - require_relative

# Contoh Modularitas ( main .rb )

```ruby
require './kalkulator.rb'

perkalian(1,6)
pembagian(6,3)
```

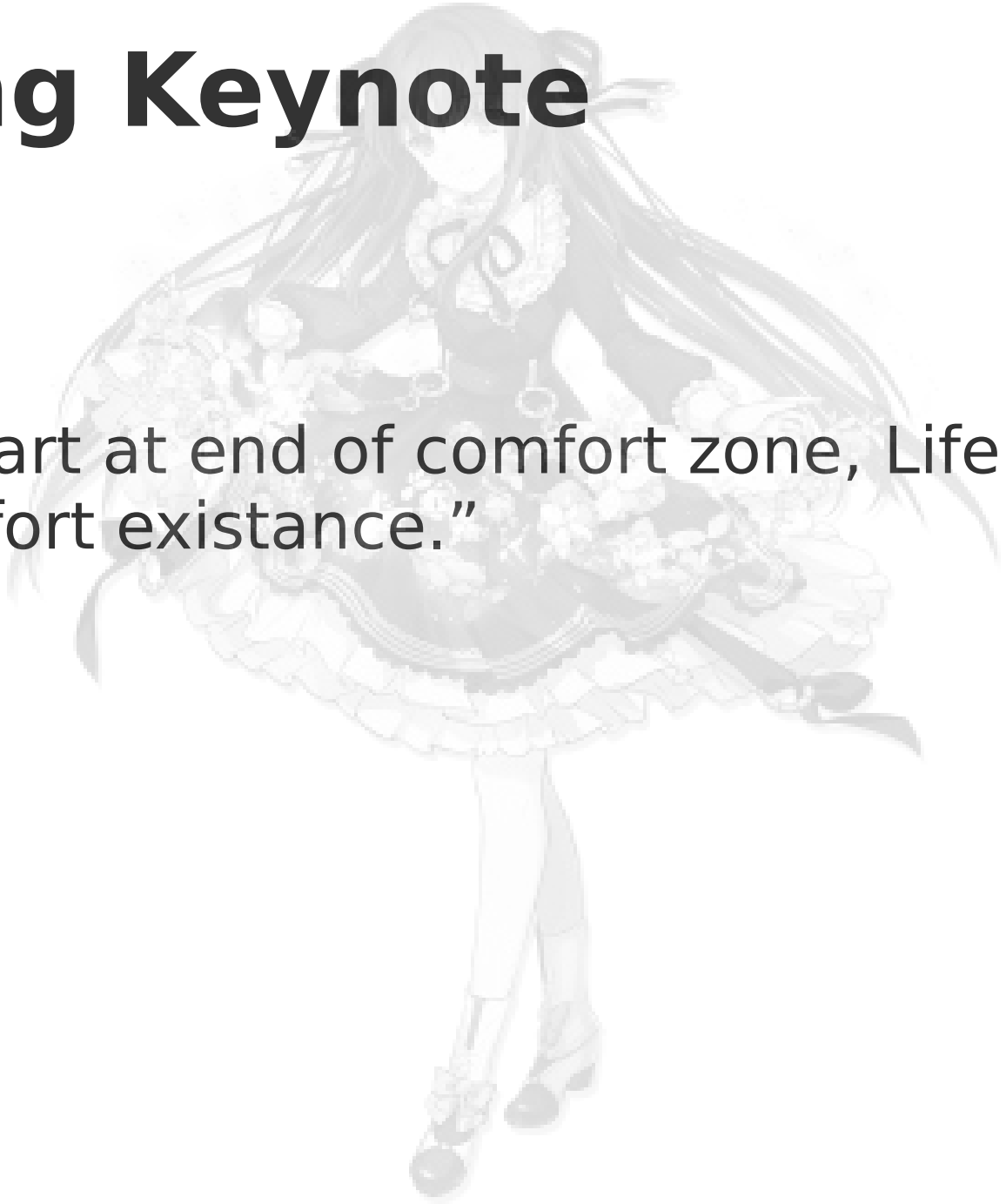
# Contoh Modularitas ( kalkulator.rb )

```ruby
# nama file : kalkulator.rb
def perkalian angka_pertama,angka_kedua
  puts (angka_pertama * angka_kedua)
end

def pembagian angka_pertama,angka_kedua
  puts (angka_pertama / angka_kedua)
end
```

# Melakukan Modularitas

- Bentuk berkas tersebut pada direktori / lokasi yang sama.
- Jalan program main.rb dan pastikan kalkulator.rb pada direktori yang sama( ./ )
- Tujuan modularitas adalah memanggil subprogram pada kalkulator.rb melalui main.rb.

# Closing Keynote



“Life start at end of comfort zone, Life is uncomfort existance.”

# 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.