

# Pertemuan5

## *Pengenalan Ruby*

### **Objektif:**

1. Mahasiswa dapat memahami sejarah perkembangan bahasa pemrograman Ruby.
2. Mahasiswa dapat mengetahui kelebihan dan kekurangan bahasa Ruby.
3. Mahasiswa dapat mengetahui fitur-fitur penting yang terdapat pada Ruby.
4. Mahasiswa dapat melakukan proses instalasi bahasa Ruby.
5. Mahasiswa mengerti konsep dan struktur bahasa pemrograman Ruby.
6. Mahasiswa mengerti konsep variabel, tipe data, dan operator pada Ruby.
7. Mahasiswa dapat membuat program sederhana menggunakan bahasa pemrograman Ruby.

## P5.1 Teori

### Sejarah dan Perkembangan Ruby

Ruby merupakan bahasa *scripting* yang memiliki *string processing* yang sangat akurat misalnya *regular expression* sehingga sangat cocok untuk administrator sistem untuk membuat *shell script* yang *powerfull*. Di pihak lain, bahasa *scripting* menawarkan pengembangan aplikasi yang cepat misalnya pembuatan aplikasi GUI, *web scripts*, *system utilities*, dan aplikasi yang membutuhkan pemrosesan *string* ataupun perhitungan yang akurat. Pencipta Ruby, Yukihiro Matsumoto (Matz), menggabungkan bagian-bagian dari bahasa-bahasa favorit beliau (Perl, Smalltalk, Eiffel, Ada dan Lisp) untuk membentuk bahasa baru yang seimbang antara pemrograman fungsional dengan pemrograman imperatif.

Sejak Ruby pertama kali dirilis ke publik pada tahun 1995, banyak *programmer* profesional dari seluruh dunia serius ikut mengembangkan Ruby. Pada tahun 2006, Ruby diterima oleh banyak orang. Dengan komunitas pengguna Ruby yang aktif di banyak kota-kota di seluruh dunia dan konferensi-konferensi beserta pertemuan Ruby terkait.

Ruby-Talk, milis utama untuk diskusi Ruby (dalam bahasa Inggris) telah mencapai kisaran 200 *email* setiap hari. TIOBE *index*, yang menghitung perkembangan bahasa-bahasa pemrograman, menempatkan Ruby pada peringkat ke 10 diantara bahasa-bahasa pemrograman di seluruh dunia. Melihat pada perkembangan ini, mereka memperkirakan, “Kesempatan Ruby memasuki peringkat atas 10 besar adalah dalam waktu setengah tahun.”

Kebanyakan dari perkembangan Ruby beratribut pada terkenalnya *software* yang ditulis dengan Ruby, terutama *framework* web Ruby on Rails. Ruby juga sepenuhnya bebas. Tidak hanya gratis, tetapi juga bebas untuk menggunakan, memodifikasi dan mendistribusikan Ruby.

### Pengenalan Bahasa Ruby

Ruby adalah bahasa pemrograman *scripting* yang berorientasi objek. Tujuan dari ruby adalah menggabungkan kelebihan dari semua bahasa pemrograman *scripting* yang ada di dunia. Ruby ditulis dengan bahasa C dengan kemampuan dasar seperti Perl dan Python.

Ruby pertama kali dibuat oleh seorang programmer Jepang bernama Yukihiro Matsumoto. Penulisan Ruby dimulai pada February 1993 dan pada Desember 1994 dirilis versi alpha dari ruby. Pada awal perkembangan Ruby, Yukihiro menulis Ruby sendiri sampai pada tahun 1996 terbentuk komunitas Ruby yang banyak berkontribusi Ruby.

Aplikasi bahasa Ruby, antara lain :

- Implementasi besar Ruby pada JRuby dan Rubinius.
- Ruby dapat diterapkan pada teknologi Asynchronous JavaScript dan XML (AJAX).
- Ruby on Rails untuk membuat framework web.

Software ruby dapat di download di situs [www.ruby-lang.org/id/downloads/](http://www.ruby-lang.org/id/downloads/)

Alasan Mengapa Banyak yang Memakai Ruby, antara lain :

- Ruby berorientasi objek
- Ruby murni merupakan bahasa pemrograman berorientasi objek
- Ruby merupakan bahasa yang dinamis
- Ruby merupakan bahasa interpreted
- Ruby bias dijalankan di banyak platform
- Ruby bersumber dari banyak sumber dan inovatif
- Ruby merupakan bahasa scripting
- Ruby memiliki garbage collector yang pintar
- Ruby bisa multi thread dan memiliki mekanisme exception
- Ruby tidak ada reserved word dan tidak punya pointer
- Ruby memiliki sintaks yang fleksibel
- Ruby punya overloading operator
- Ruby kaya akan library dan memiliki debugger

### **Kelebihan dan Kekurangan Ruby**

Adapun kelebihan dan kekurangan yang dimiliki oleh Bahasa Pemrograman Ruby adalah sebagai berikut.

#### **Kelebihan :**

- Sintaks sederhana.
- Memiliki Exception Handling yang baik.
- OOP.
- Single inheritance.
- Didukung oleh OS Linux, Windows, MacOS X, OS/2, BeOs, dan Unix.
- Merupakan bahasa pemrograman scripting yang berorientasi objek.

- Memiliki garbage collector yang secara otomatis akan menghapus informasi tak terpakai dari memori.

### **Kelemahan:**

- Multithreading. Implementasi thread di ruby masih berupa green thread, bukan native thread. Hal ini membuat aplikasi GUI (desktop) dengan background thread tidak mungkin diimplementasikan di ruby.
- Virtual Memory, ruby masih fully interpreted sehingga program ruby cenderung lebih lambat.
- Spek. saat ini spesifikasi ruby (syntax, behaviour, dll) adalah implementasi ruby yang asli dari matz.
- IDE. Saat ini kualitas IDE untuk ruby masih jauh daripada .net dan java. Tapi dengan bermunculnya IDE ruby yang dibuat dengan java.net, kondisinya agak berubah. Tapi karena ruby bahasa yang sangat dinamis, sulit untuk bisa mendapatkan informasi secara lengkap mengenai struktur sebuah program ruby secara statis.

### **Fitur-fitur Pada Ruby**

Pertama kali, Matz melihat bahasa-bahasa lain untuk mencari sintaks yang ideal. Terkenang pencariannya, Matz berkata, “Saya mau bahasa *scripting* yang lebih hebat daripada Perl dan lebih berorientasi obyek daripada Ruby.” Di Ruby, semua adalah obyek. Setiap informasi dan kode bisa diberi *property* dan *action*. Pemrograman berorientasi obyek memanggil *property* dengan nama variabel *instant* dan *action*, yang disebut sebagai metode. Pendekatan murni berorientasi obyek terutama terlihat pada demonstrasi sedikit kode yang diberikan pada number.

Ruby dianggap sebagai bahasa yang fleksibel, karena bagian-bagian dari Ruby bisa diubah-ubah dengan bebas. Bagian-bagian yang esensi di Ruby bisa dihapus maupun didefinisikan ulang. Bagian-bagian yang sudah ada bisa ditambahkan. Ruby mencoba untuk tidak membatasi *programmer*.

Ruby kaya fitur, antara lain sebagai berikut:

- Ruby merupakan bahasa interpreter.
- Ruby memiliki sintaks yang sederhana, mudah dipelajari dan dipahami.
- Ruby memiliki fitur-fitur yang menangani *exception*, seperti Java atau Ruby, untuk mempermudah menangani *error*.

- Ruby menyediakan *mark-and-sweep garbage collector* untuk semua obyek Ruby. Tidak perlu me-maintain *reference count* pada *library extension*.
- Menulis *extension* C di Ruby lebih mudah daripada di Perl ataupun di Ruby, dengan API yang elegan untuk memanggil Ruby dari C. Ini termasuk memanggil Ruby *embedded* di software, untuk digunakan sebagai bahasa *scripting*. Interface SWIG juga tersedia.
- Ruby bisa *load library extension* secara dinamis jika Sistem Operasi mengijinkan.
- Ruby menyediakan fitur OS *threading* yang *independent*. Maka, untuk semua *platform* dimana Ruby berjalan, kita juga punya *multithreading*, terlepas dari apakah Sistem Operasi mendukung *multithreading* atau tidak, bahkan pada MS-DOS sekalipun.
- Ruby sangat *portable*: Ruby kebanyakan dikembangkan di GNU/Linux, tetapi juga berjalan di banyak tipe UNIX, Mac OS X, Windows 95/98/Me/NT/2000/XP, DOS, BeOS, OS/2, dan lain-lain.

## Proses Instalasi Ruby

Proses instalasi Ruby pada Windows, langkah-langkah sebagai berikut :

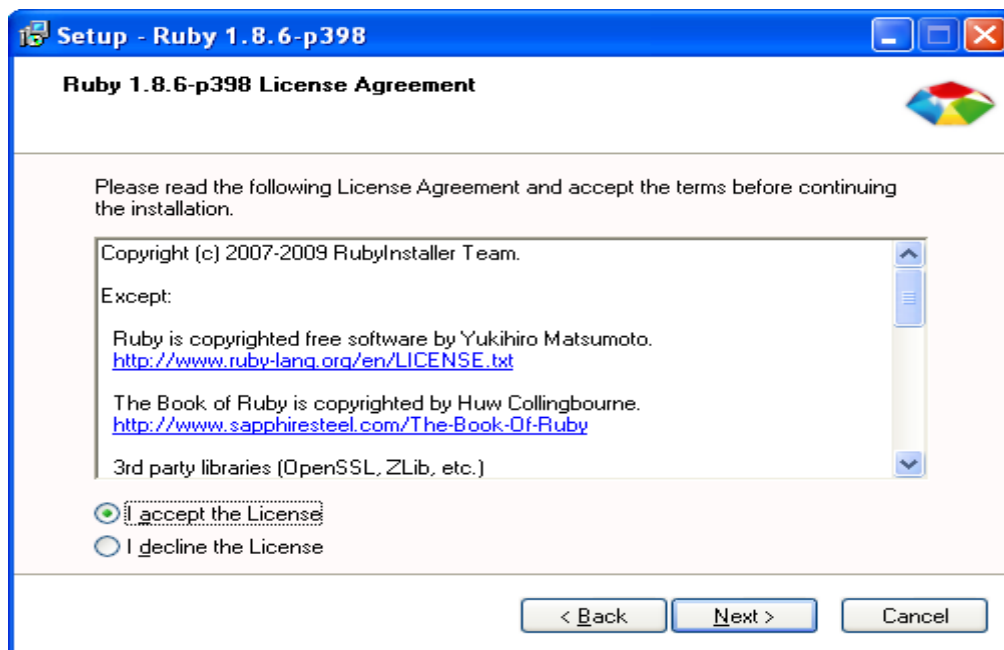
- 1) Pilih software Ruby yang diinginkan, contoh memakai Ruby versi 1.8.6 yang dapat di download pada situs [www.ruby-lang.org/id/downloads/](http://www.ruby-lang.org/id/downloads/)



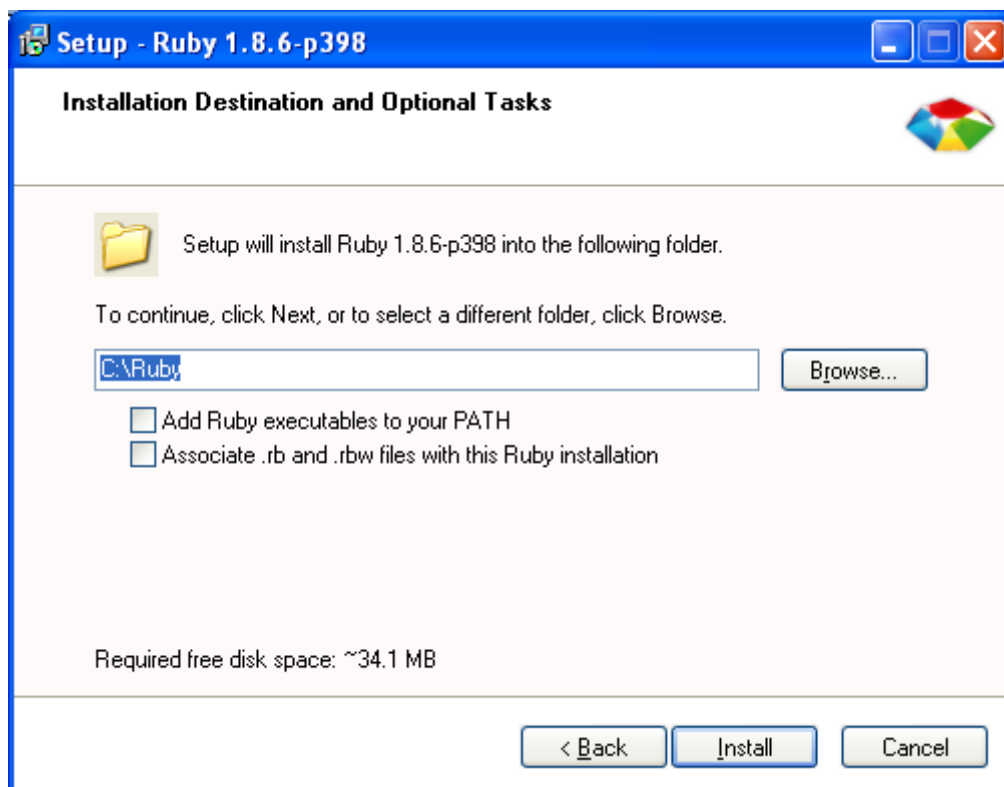
- 2) Klik software Ruby, lakukan peng-instalan pada computer dan ikuti langkah selanjutnya :



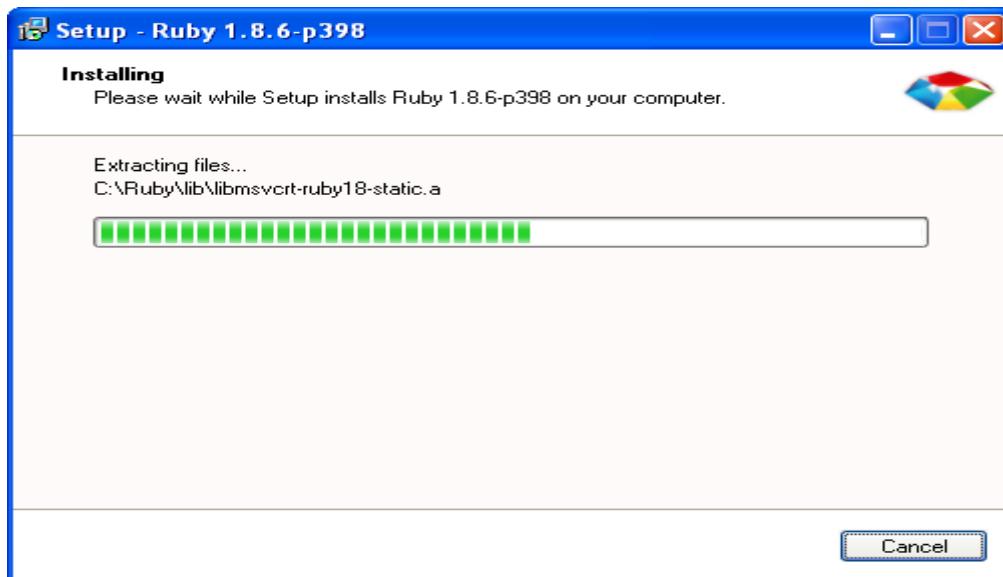
- 3) Pilih konfirmasi persetujuan, lalu klik tombol next :



- 4) Pilih direktori tujuan untuk tempat menyimpan program ruby, lalu klik tombol Install:



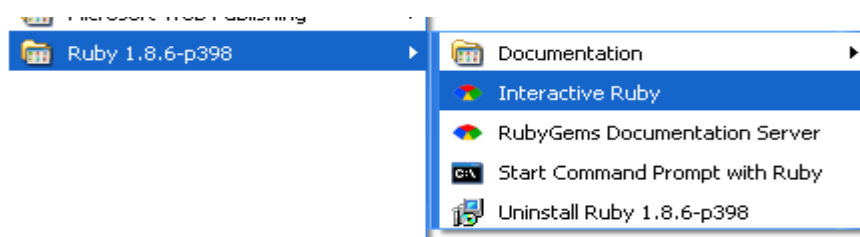
- 5) Setelah menekan tombol install, tunggu beberapa menit selama proses instalasi berlangsung dan tekan finish, ikuti petunjuk selanjutnya :



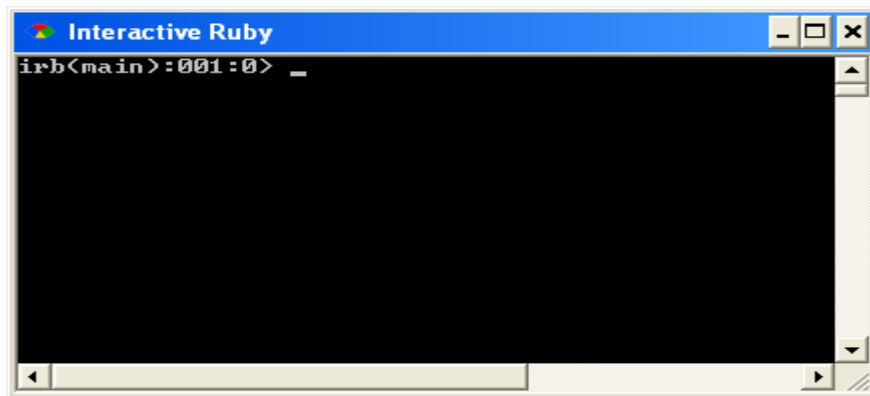
- 6) Selesai instalasi tekan Finish, komputer Anda telah ada Ruby, khususnya versi 1.8.6 pada folder C:\Ruby.

### Cara Menjalankan Program Ruby

- Klik tombol start Program Ruby 1.8.6 lalu memilih irb (interactive ruby) dengan tampilan sebagai berikut :

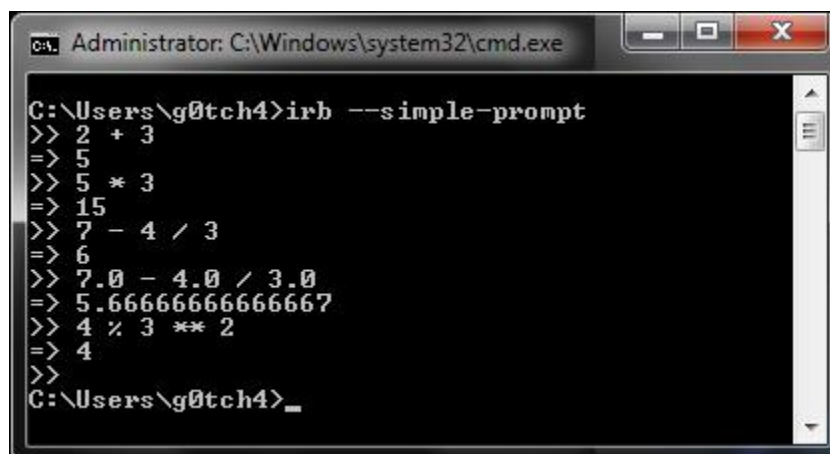


- Pada window irb, merupakan editor untuk menuliskan program ruby yang diinginkan dengan tampilan sebagai berikut :



## Dasar Pemrograman Ruby

Sebagai bahasa *scripting* yang berbasis interpreter, Ruby menawarkan modus interaktif, yakni *Interactive Ruby* yang disingkat dengan irb. Untuk masuk ke modus interaktif Ruby, cukup mengetikkan irb pada *command prompt*, untuk keluar cukup menekan tombol Ctrl-D atau ketikkan “exit”. Dapat juga menambahkan opsi “—simple-prompt” pada irb agar bentuk prompt-nya berubah menjadi >>.



Contoh Penulisan *Sintax* Ruby pada *Command Prompt*

Di samping itu, tentunya program Ruby juga dapat disimpan dalam file berextension ‘.rb’. Sama halnya seperti Ruby, modus interaktif Ruby juga dapat digunakan sebagai kalkulator untuk perhitungan sederhana. Kemampuan menghitung Ruby tidak kalah jika dibandingkan dengan Ruby. Konon, Ruby memiliki kecepatan eksekusi program yang lebih cepat dibandingkan dengan Ruby.



## Aturan penamaan variabel pada Ruby

Karena Ruby bersifat *dynamic-typing*, Kita tidak perlu mendeklarasikan tipe dan variable yang ingin kita gunakan seperti pada bahasa C. Kita cukup memasukkan nilai ke variable yang ingin kita pakai. Aturan penamaan variabel sama dengan aturan penamaan variabel pada umumnya, yakni tidak boleh dimulai dengan angka dan tidak memakai *keyword* penting dari bahasa tersebut.

Berikut contoh deklarasi variabel:

Contoh variabel yang benar:

```
x = 888
```

```
ini_variabel_string = "halo"
```

Contoh variabel yang tidak benar:

```
8x = 888
```

```
while = "halo" # while adalah reserved keyword
```

## Konstanta

Konstanta tidak lain adalah sebuah variabel yang isinya tetap (tidak berubah). Konstanta di Ruby dideklarasikan dengan huruf kapital pada huruf pertama. Suatu konstanta tetap dapat diubah nilainya. Pemberian status konstanta pada suatu variabel hanya akan memberikan suatu peringatan apabila kita mengubah isi konstanta tersebut.

## Fake Keyword Parameters

Ruby tidak memiliki keyword parameters atau parameter bernama, seperti Python. Tetapi, ini bisa dimanipulasi (fake) dengan cara menggunakan symbol dan hash. Ruby on Rails, satu diantara banyak aplikasi yang dibuat dengan Ruby, banyak menggunakan trik ini. Contoh:

```
def keluaran( params )
  params
end

keluaran( :param_satu => 10, :param_dua => 42 )
# => { :param_satu=>10, :param_dua=>42 }
```

Ini dikarenakan Ruby otomatis mengubah parameter yang diberikan tersebut menjadi bentuk hash. (meskipun pemanggilannya tanpa menggunakan kurung kurawal yang merupakan sintaks hash).

### Nilai True bersifat Universal

Di Ruby, semua (kecuali nil dan false) dianggap true. Di C, Python dan banyak bahasa lain, 0 dan mungkin juga nilai-nilai lain, seperti list yang kosong, dianggap false. Perhatikan kode Python berikut (contoh berikut juga berguna untuk bahasa-bahasa lain):

```
# di Python
if 0:
    print "0 is true"
else:
    print "0 is false"
```

Ini akan print 0 jika false . Kode yang sama di Ruby:

```
# di Ruby
if 0
    puts "0 is true"
else
    puts "0 is false"
end
```

### Access Modifier Berlaku Sampai Akhir Scope

Contoh kode Ruby berikut ini :

```
class KelasSaya
  private
  def metode; true;
end
  def metode_lain; false;
end
end
```

Tetapi tidak demikian. Access modifier private bersambung terus sampai akhir scope, atau sampai access modifier lain muncul, apapun yang muncul lebih awal. Secara default, metode bersifat public:

```

class KelasSaya
  # Sekarang metode adalah public
  def metode;
  true;
end

  private

  # metode_lain adalah private
  def metode_lain;
  false;
end
end

```

Public, private dan protected benar-benar merupakan metode, sehingga mereka bisa menerima parameter. Jika melewatkan symbol pada satu dari parameter, maka visibility metode diubah.

### Akses Metode

Di Java, public berarti metode bisa diakses oleh siapa saja. protected berarti instance kelas, instance dari kelas-kelas turunan, dan instance dari kelas-kelas package yang sama dapat mengakses, tetapi tidak untuk yang lain, dan private berarti tidak ada yang dapat mengakses metode kecuali instance kelas.

Di Ruby, public secara natural adalah public. private berarti metode hanya bisa diakses ketika metode bisa dipanggil tanpa obyek penerima yang eksplisit. Hanya **self** yang boleh menjadi receiver pemanggilan metode private.

### Kelas Bersifat Terbuka

Kelas Ruby bersifat terbuka. Anda bisa membuka Class, lalu menambahkan ke dalam Class, dan menggantinya kapan saja. Bahkan kelas yang termasuk class inti, seperti kelas Fixnum atau bahkan kelas Object, induk dari semua obyek di Ruby. Ruby on Rails mendefinisikan banyak metode yang berhubungan dengan waktu ke dalam kelas Fixnum. Perhatikan kode berikut:

```

class Fixnum
  def hours
    self * 3600 # total detik dalam satu jam adalah 3600
  end
end

```

```

alias hour hours
end

# 14 hours from 00:00 January 1st
# jadi 14 jam dari jam 00:00 pada tanggal 1 Januari
Time.mktime(2007, 01, 01) + 14.hours # => Sun Jan 01 14:00:00

```

## Nama Metode Deskriptif dan Menarik

Di Ruby, metode boleh diakhiri dengan tanda tanya ataupun tanda seru. Pengaturan nama adalah, metode-metode yang menjawab pertanyaan (seperti `Array#empty?` mengembalikan nilai **true** jika obyek penerima yaitu `Array` ternyata kosong) diakhiri dengan tanda tanya. Kemudian, metode-metode yang berpotensi untuk membahayakan (seperti metode yang mengganti **self** atau argumen, `exit!` dan lain sebagainya) maka pengaturannya diakhiri dengan tanda seru. Tetapi, semua metode yang mengganti argument tidak diakhiri dengan tanda seru. `Array#replace` mengganti isi array dengan isi array lain. Karena tidak masuk akal kalau ada metode sedemikian rupa **tidak** mengganti `self`.

## Metode Singleton

Metode singleton merupakan metode-metode yang basisnya per obyek. Singleton hanya tersedia pada obyek yang Anda definisikan, jadi metode tersebut tidak tersedia pada obyek-obyek yang lain dari kelas yang sama.

```

class Mobil
  def inspect
    "Mobil murah"
  end
end

porsche = Mobil.new
porsche.inspect # => Mobil murah

def porsche.inspect
  "Mobil mahal"
end

porsche.inspect # => Mobil mahal

# Sementara obyek-obyek yang lain tidak terpengaruh

```

```
mobil_lain = Mobil.new
mobil_lain.inspect # => Mobil murah
```

### Metode `method_missing`

Ruby tidak menyerah kalau Ruby tidak mendapatkan metode yang bisa menanggapi message tertentu. Ruby akan memanggil metode `method_missing` dengan nama metode yang Ruby tidak ditemukan beserta daftar parameternya. Secara default, `method_missing` membangkitkan exception `NameError`, tetapi Anda bisa mendefinisikan ulang Exception tersebut supaya lebih sesuai dengan aplikasi yang Anda buat, dan banyak library yang melakukan hal yang sama.

### Melewatkan Message, Bukan Pemanggilan function

Pemanggilan metode adalah sungguh-sungguh merupakan **message** ke obyek lain:

```
# Penambahan ini
1 + 2
# adalah sama dengan penambahan ini
1.+(2)
# juga sebetulnya sama dengan penambahan ini :
1.send "+", 2
```

### *Input Output*

Untuk meminta *input*-an dari *user*, kita menggunakan perintah `gets`. Sedangkan untuk *output* ke layar monitor, kita dapat menggunakan `puts`, `print` maupun `printf`.

```
>> puts "Halo, apa kabar ?"
Halo, apa kabar?
=> nil
>> print "Halo, apa kabar ?"
Halo, apa kabar ?=> nil
>> nama = gets
g0tch4
=> "Eric\n"
>> printf "Nama saya %s", nama
```

```
printf "Nama saya %s", nama
```

```
Nama saya g0tch4
```

```
=> nil
```

Adapun perbedaan antara `puts`, `print` dan `printf` yakni di mana `puts` akan menambahkan karakter *newline* (`'\n'`) pada akhir string dan parameternya harus *string*, sedangkan `print` hanya mencetak string tanpa menambahkan karakter *newline*, `printf` sama dengan `print`; bedanya `printf` mengenal formatting seperti `%s` untuk string, `%f` untuk float, `%d` untuk integer, dan seterusnya. Perintah `printf` ini sama dengan perintah `printf` di bahasa C.

## Struktur Program

Program Ruby umumnya juga memakai indentasi seperti bahasa Ruby. Akan tetapi indentasi tidak mutlak harus dilakukan, karena Ruby menggunakan *keyword end* untuk menandakan akhir dari suatu bagian program.

## Lain-lain

Mungkin pada beberapa contoh di atas, sering melihat tulisan `nil`. `Nil` berarti suatu objek di Ruby sama dengan `NULL` di bahasa C. `Nil` berarti hasil eksekusi perintah tersebut tidak mengembalikan objek apapun alias `nil` (tidak memiliki *return value*). Misalkan perintah `puts` hanya mencetak *string* ke layar dan tidak mengembalikan objek apapun untuk disimpan alias `nil`. Akan tetapi lainnya halnya dengan `a = "halo"` akan mengembalikan objek *String* `"halo"` yang akan disimpan dalam variabel `a`. Untuk komentar pada program Ruby, Kita dapat menggunakan tanda `#`. Untuk lebih dari satu *statement* pada satu baris, Kita dapat menggunakan pemisah `;`. Sedangkan untuk *statement* yang lebih dari satu baris, Kita dapat menggunakan tanda ```.

```
>> a = 1#Variabel a berisi 1
```

```
=> 1
```

```
>> print "Halo "; puts " apa  
kabar ?"
```

```
Halo apa kabar ?
```

```
=> nil
```

```
>> b = 1 + 3 + 5 \
```

```
?> + 7 + 9
```

```
=> 25
```

## Tipe Data Dasar

Setelah berkenalan dengan dasar-dasar interpreter Ruby, selanjutnya dibahas beberapa tipe data dasar yang disediakan Ruby yang tentunya merupakan instansi dari kelas–kelas mengingat Ruby adalah bahasa berorientasi objek yang murni. Di samping itu, akan dibahas beberapa metode yang umum dari kelas– kelas tersebut.

### 1. Angka

Ruby dapat menangani angka baik yang bertipe integer maupun float. Untuk tipe data Integer di Ruby, kelas Integer dibagi dalam dua kelas yakni FixNum dan BigNum. Angka dengan batas -230 sampai 230–1 tergolong dalam kelas FixNum; apabila suatu angka telah melampaui batas tersebut, maka akan digolongkan dalam kelas BigNum. Karena Ruby bersifat *dynamic–typing*, Kita tidak perlu melakukan konversi dari FixNum ke BigNum karena konversi akan dilakukan secara otomatis. Sedangkan untuk angka yang bertipe float, Ruby akan menganggap objek angka tersebut merupakan instansi dari kelas Float.

Seperti bahasa pemrograman umumnya, pada Ruby dapat menggunakan prefiks (awalan) untuk menandakan arti angka tersebut, misalya untuk menyatakan bilangan negatif, 0 untuk bilangan oktal , 0b untuk bilangan biner dan 0x untuk bilangan heksadesimal serta e untuk bilangan eksponensial. Untuk mempermudah penulisan suatu angka dengan nilai yang sangat besar, Kita dapat membubuhkan karakter \_ pada penulisan angka (karakter \_ tidak akan disimpan, hanya untuk membantu saja).

```
>> a = 2
=> 2
>> a.class
=> FixNum
>> a = a ** 31
=> 2147483648
>> a.class
=> BigNum
>> 188_888_000_000
=> 188888000000
>> 0x6AF
=> 1711
>> 1.89e+18
=> 1.89e+18
```

```
>> b = 1.4
=> 1.4
>> b.class
=> Float
```

## 2. String

Tipe data String di Ruby sama dengan tipe data String pada bahasa pemrograman lain umumnya. Untuk membuat tipe data String, kita dapat menggunakan kutip satu ‘ ataupun kutip dua “. Adapun perbedaan di antara keduanya, yakni di mana objek String yang dibuat dengan kutip dua “ akan mengerti karakter khusus seperti ‘\n’, ‘\r’, ‘\b’ ,dsb. ; sedangkan karakter dengan objek String yang dibuat dengan kutip satu ‘ tidak bisa menerjemahkan karakter khusus di atas.

Perhatikanlah contoh berikut:

```
>> a = "Hello\n"
=> "Hello\n"
>> print a
Hello
=> nil
>> b = 'Hello\n'
=> "Hello\n"
>> print b
Hello\n=> nil
```

Kelas String kaya akan metode-metode yang powerful. Kita dapat mencoba beberapa di antaranya seperti berikut ini:

1. *Length* : untuk mengetahui panjang suatu string.
2. *Capitalize* : untuk mengubah huruf pertama pada awal kalimat menjadi huruf kapital.
3. *Downcase* : mengubah string menjadi huruf kecil.
4. *Uppcase* : mengubah string menjadi huruf besar.
5. *Swapcase* : mengubah objek string dengan huruf kecil diubah menjadi huruf kapital dan sebaliknya.
6. *Strip* : membuang karakter *whitespace* di awal dan akhir string.
7. *Reverse* : membalikkan string.
8. *Include ? str* : mengembalikan true jika substring str terdapat dalam string dan false jika tidak ada.



9. *Chop* : membuang karakter terakhir dari string.

Salah satu hal yang unik dari Ruby adalah Kita dapat menambahkan tanda '!' di akhir metode untuk menandakan metodenya bersifat destruktif di mana hasil metode tersebut berdampak langsung pada objeknya.

```
>> "Hello".length
=> 5
>> "hello".capitalize
=> "Hello"
>> "HELlo".downcase
=> "hello"
>> "HeLlo".upcase
=> "HELLO"
>> "hElLo".swapcase
=> "HeLlO"
>> "hello".reverse
=> "olleh"
>> " hello ".strip
=> "hello"
>> "helloo".chop
=> "hello"
>> "hello".include? "h"
=> true
>> a = " Hello "
=> " Hello "
>> a.strip
=> "Hello"
>> a
=> " Hello "
>> a.strip!
=> "Hello"
>> a
=> "Hello"
```

## Operator pada Ruby

### Operator adalah Syntactic Sugar

Kebanyakan operator di Ruby hanyalah syntactic sugar. Maksudnya syntactic sugar adalah penyingkatan penulisan kode. Dalam hal operator-operator itu, mereka sebenarnya hanyalah pemanggilan metode saja, tentunya dengan peraturan tertentu supaya jenjang precedence tetap dituruti. Contohnya, Anda bisa meng-override metode `+` milik kelas `Fixnum`:

```
class Fixnum
  # Sebenarnya Anda bisa melakukan ini,
  # tetapi tolong jangan lakukan ini
  def +(other)
    self - other
  end
end
```

Anda tidak membutuhkan operator `++`, dan seterusnya. Kecuali operator-operator dibawah ini bukan syntactic sugar. Operator-operator dibawah ini bukan metode dan tidak dapat didefinisikan ulang:

`=, ..., ..., !, not, &&, and, ||, or, !=, !~, ::`

Tambahan, `+=`, `*=` dan lain sebagainya hanyalah singkatan untuk `var = var + var_lain`, `var = var * var_lain`, dan seterusnya tidak dapat didefinisikan ulang.

### Operator OR pada Ruby

Operator OR pada bahasa pemrograman Ruby dapat digunakan tidak hanya saat *conditional if*, tetapi juga dapat digunakan untuk *assignment* variabel. Berikut ini adalah sedikit penjelasan penggunaan operator OR (`||`) pada bahasa pemrograman Ruby.

1. `c = a || b` atau `c = a or b`

Maksud sintaks diatas adalah jika variabel `a` nil atau false, maka variabel `c` sama dengan variabel `b`.

2. `c ||= 3`

Sintaks diatas adalah kependekan dari sintaks poin pertama. Dengan kata lain, sintaks `c ||= 3` sama dengan `c = c || 3` yang berarti jika `c` belum terdefinisi maka akan diisi dengan nilai 3.

## Contoh Program Sederhana Menggunakan Ruby

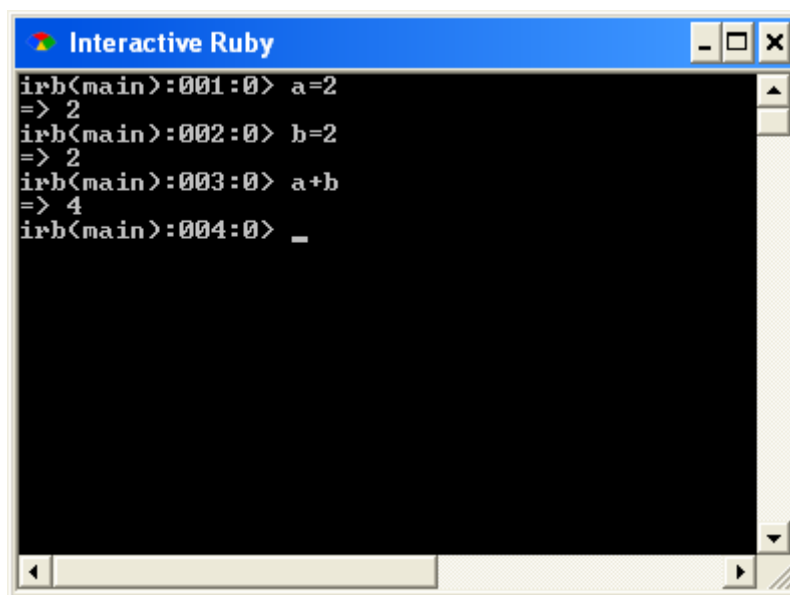
Pada contoh program menggunakan skrip berarti menyusun statemen-statemen menjadi sebuah satu kesatuan file ruby. Dengan membuat skrip berarti kita melakukan kompilasi file ruby dengan bantuan Interpreter dari ruby lewat Command Prompt (Windows) atau Terminal (Linux/Unix). Contohnya,

### Program 1.

```
# Program ruby 1  
irb> a=2  
irb> b=2  
irb> a+b
```

Tulis skrip diatas pada Text Editor interactive ruby, kemudian lakukan kompilasi dengan menekan enter dan secara otomatis output akan tampil.

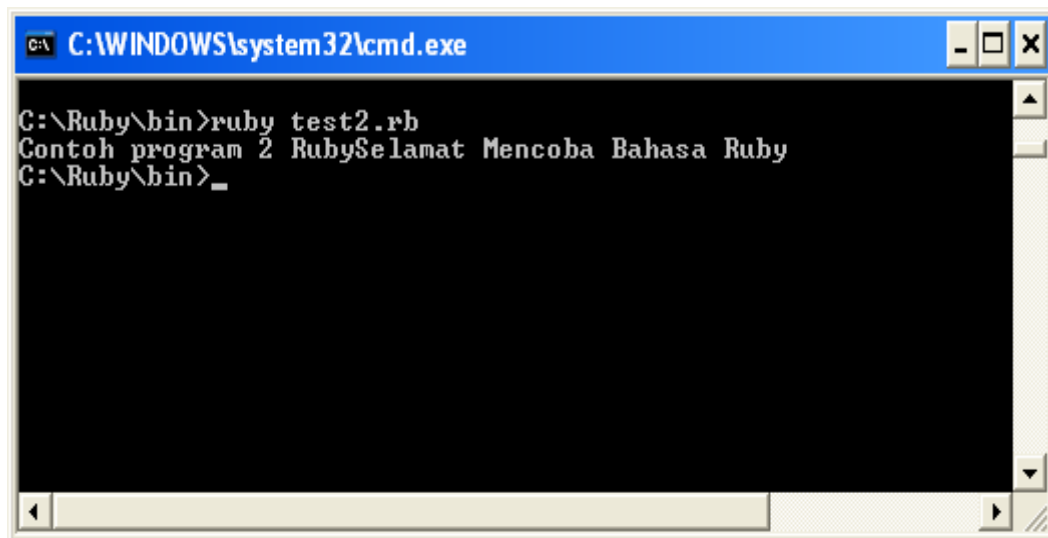
### Output :



### Program 2.

```
# Program ruby kedua  
print "Contoh program 2 Ruby"  
print "Selamat Mencoba Bahasa Ruby"
```

Tulis skrip diatas pada notepad lalu simpan file dengan nama test2.rb lalu compile dengan cara ketik ruby test2.rb pada command prompt (cmd) dengan output sebagai berikut.



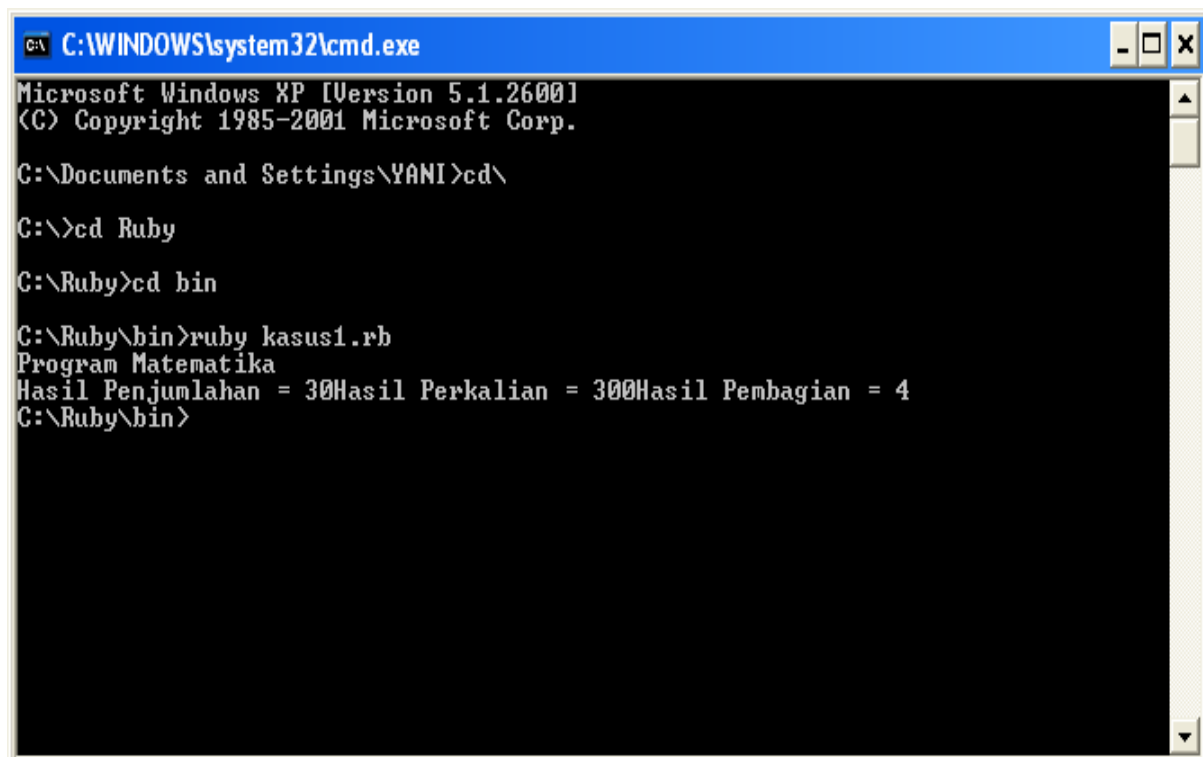
```
C:\WINDOWS\system32\cmd.exe

C:\Ruby\bin>ruby test2.rb
Contoh program 2 RubySelamat Mencoba Bahasa Ruby
C:\Ruby\bin>
```

## P5.2 ContohKasus

### Contoh Kasus 1

Pada contoh kasus yang pertama yaitu membuat program matematika terdiri dari penjumlahan, perkalian, dan pembagian. Untuk menampilkan hasil penjumlahan, perkalian, dan pembagian dua buah bilangan menggunakan tipe data dan operator pada ruby. Output yang akan ditampilkan adalah sebagai berikut :



```
C:\WINDOWS\system32\cmd.exe

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\YANI>cd\

C:\>cd Ruby

C:\Ruby>cd bin

C:\Ruby\bin>ruby kasus1.rb
Program Matematika
Hasil Penjumlahan = 30Hasil Perkalian = 300Hasil Pembagian = 4
C:\Ruby\bin>
```

Langkah-langkah pengerjaan adalah sebagai berikut :

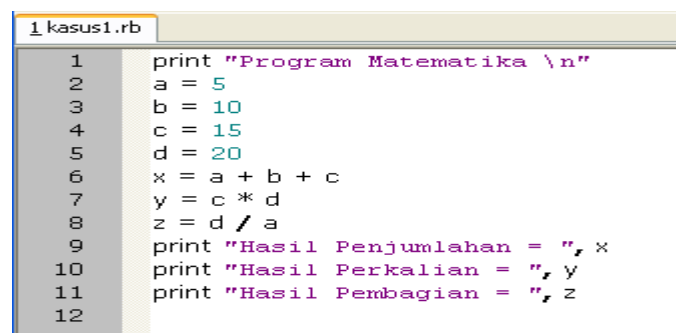
1. Klik tombol start Program Ruby 186-26 lalu pilih SciTe dengan tampilan sebagai berikut :



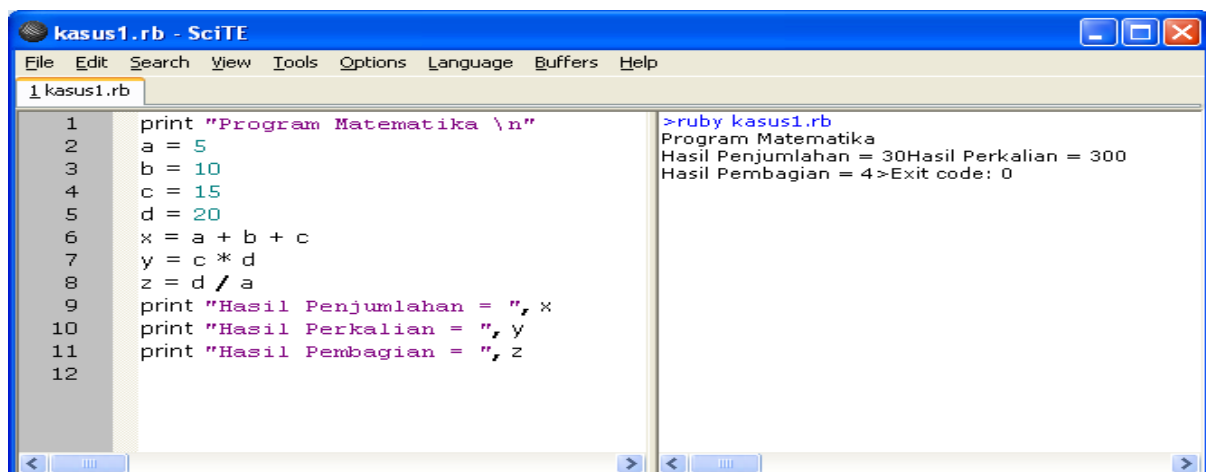
2. Klik Menu File -> New lalu ketikkan listing program sebagai berikut.

```
print "Program Matematika \n"
a = 5
b = 10
c = 15
d = 20
x = a + b + c
y = c * d
z = d / a
print "Hasil Penjumlahan = ", x
print "Hasil Perkalian = ", y
print "Hasil Pembagian = ", z
```

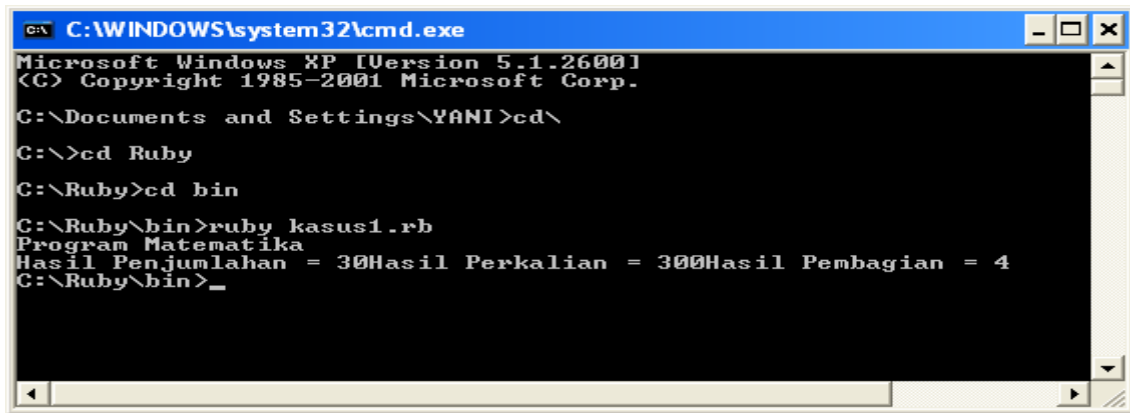
3. Setelah selesai mengetikkan code, langkah selanjutnya menyimpan file tersebut dengan cara klik menu File -> Save As. Masukkan nama file dengan nama kasus1.rb



4. Setelah itu menjalankan program dengan cara menekan tombol F5.



5. Selain itu kita juga dapat menjalankan program pada command prompt dengan cara mengetikkan `C:\Ruby\bin> ruby kasus1.rb`



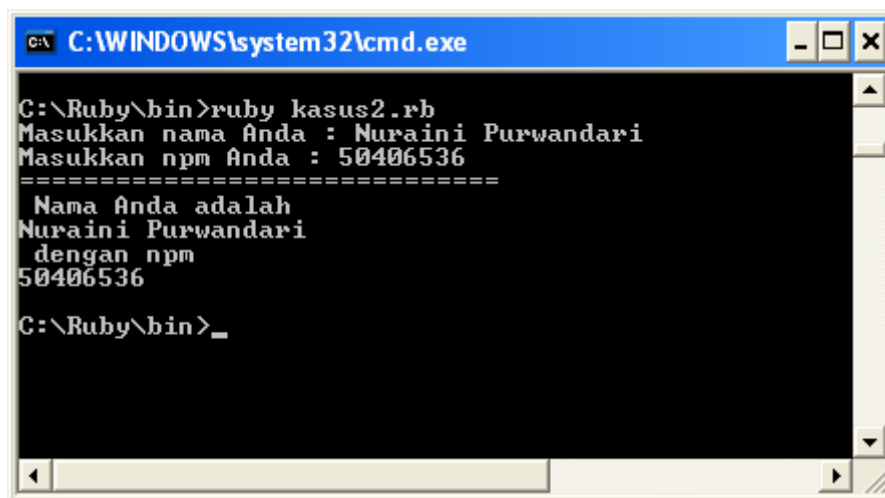
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\YANI>cd\
C:\>cd Ruby
C:\Ruby>cd bin
C:\Ruby\bin>ruby kasus1.rb
Program Matematika
Hasil Penjumlahan = 30Hasil Perkalian = 300Hasil Pembagian = 4
C:\Ruby\bin>_
```

6. Apabila tidak ada error maka program yang kita compile berhasil.

### Contoh Kasus 2

Pada contoh kasus yang kedua yaitu membuat program menginput dan menampilkan kalimat menggunakan bahasa ruby. Output yang akan ditampilkan adalah sebagai berikut :

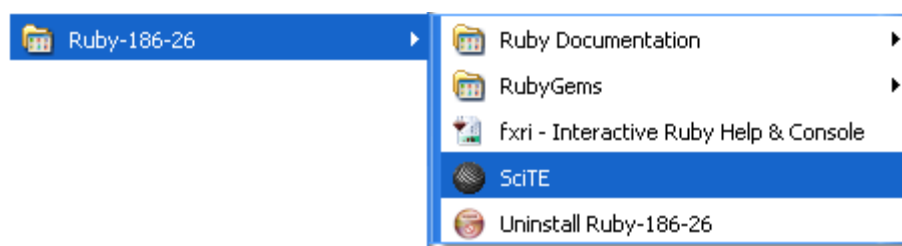


```
C:\WINDOWS\system32\cmd.exe

C:\Ruby\bin>ruby kasus2.rb
Masukkan nama Anda : Nuraini Purwandari
Masukkan npm Anda : 50406536
=====
Nama Anda adalah
Nuraini Purwandari
dengan npm
50406536
C:\Ruby\bin>_
```

Langkah-langkah pengerjaan adalah sebagai berikut :

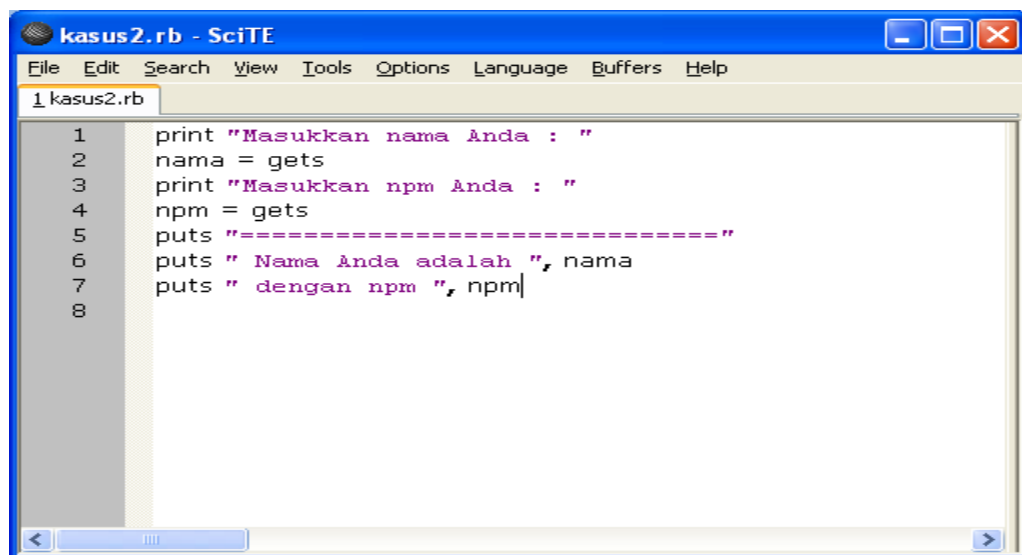
- 1). Klik tombol start Program Ruby 186-26 lalu pilih SciTe dengan tampilan sebagai berikut :



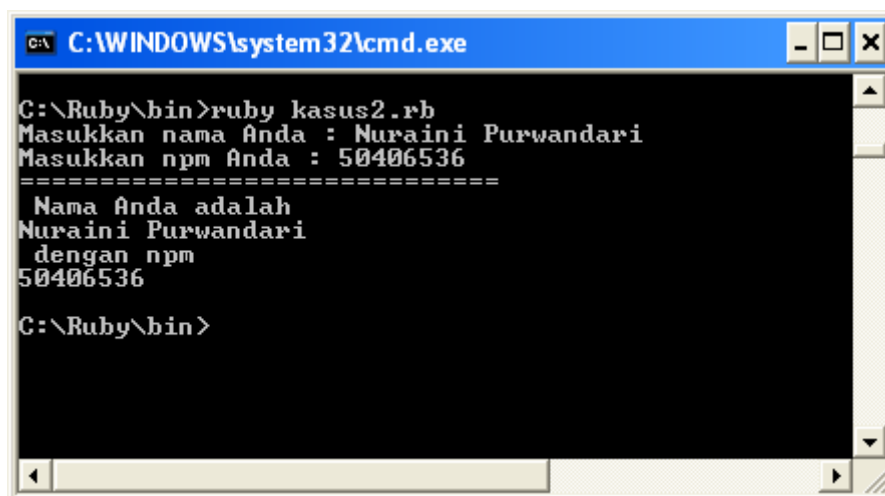
2). Klik Menu File -> New lalu ketikkan listing program sebagai berikut.

```
print "Masukkan nama Anda : "  
nama = gets  
print "Masukkan npm Anda : "  
npm = gets  
puts "=====  
puts " Nama Anda adalah ", nama  
puts " dengan npm ", npm|
```

3). Setelah selesai mengetikkan code, langkah selanjutnya menyimpan file tersebut dengan cara klik menu File -> Save As. Masukkan nama file dengan nama kasus2.rb



- Selain itu kita juga dapat menjalankan program pada command prompt dengan cara mengetikkan C:\Ruby\bin> ruby kasus2.rb



6). Apabila tidak ada error maka program yang kita compile berhasil.

## P5.3 Latihan

### Latihan 1

Berikut ini merupakan program yang menggunakan bahasa ruby versi 2.6 untuk mencari hasil penjumlahan dua bilangan serta menginput dan menampilkan nama, npm, mata praktikum. Pada code editor di ruby 186 ketikkan program berikut.

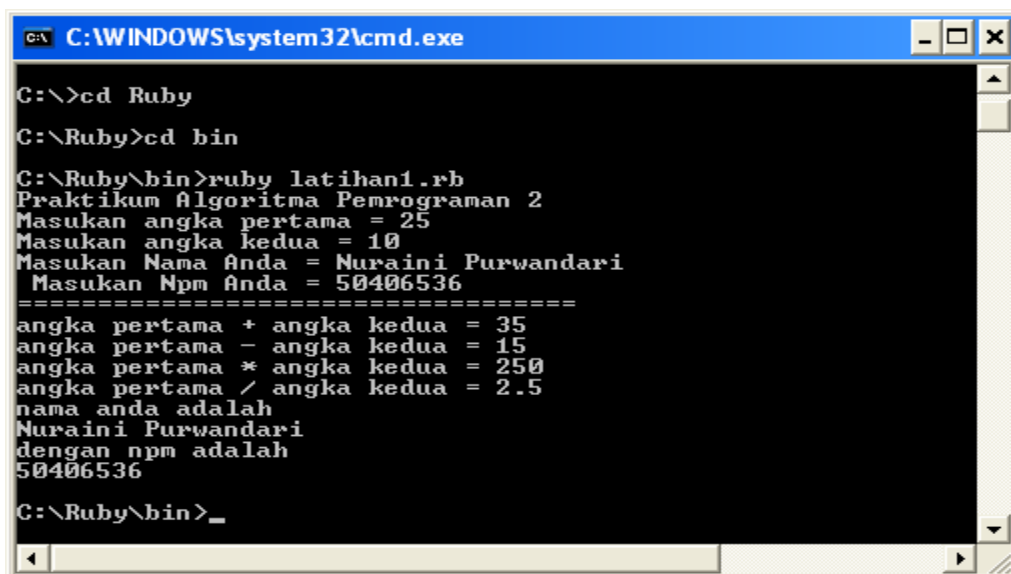
(Lengkapi kode program berikut dengan mengisi titik-titik yang berwarna merah)

```
praktikum = "Algoritma Pemrograman 2" // pendeklarasian variable praktikum

puts "Praktikum Algoritma Pemrograman 2"
print "Masukan angka pertama = "
.....
print "Masukan angka kedua = "
b=gets
print "Masukan Nama Anda = "
.....
print " Masukan Npm Anda = "
npm = gets
puts "=====
puts "angka pertama + angka kedua = #{a.to_i+b.to_i}"
.....
puts "angka pertama * angka kedua = #{a.to_i*b.to_i}"
.....
puts "nama anda adalah ", nama
.....
```

(Save program diatas dengan nama file latihan1.rb)

### TAMPILAN OUTPUT PROGRAM LATIHAN

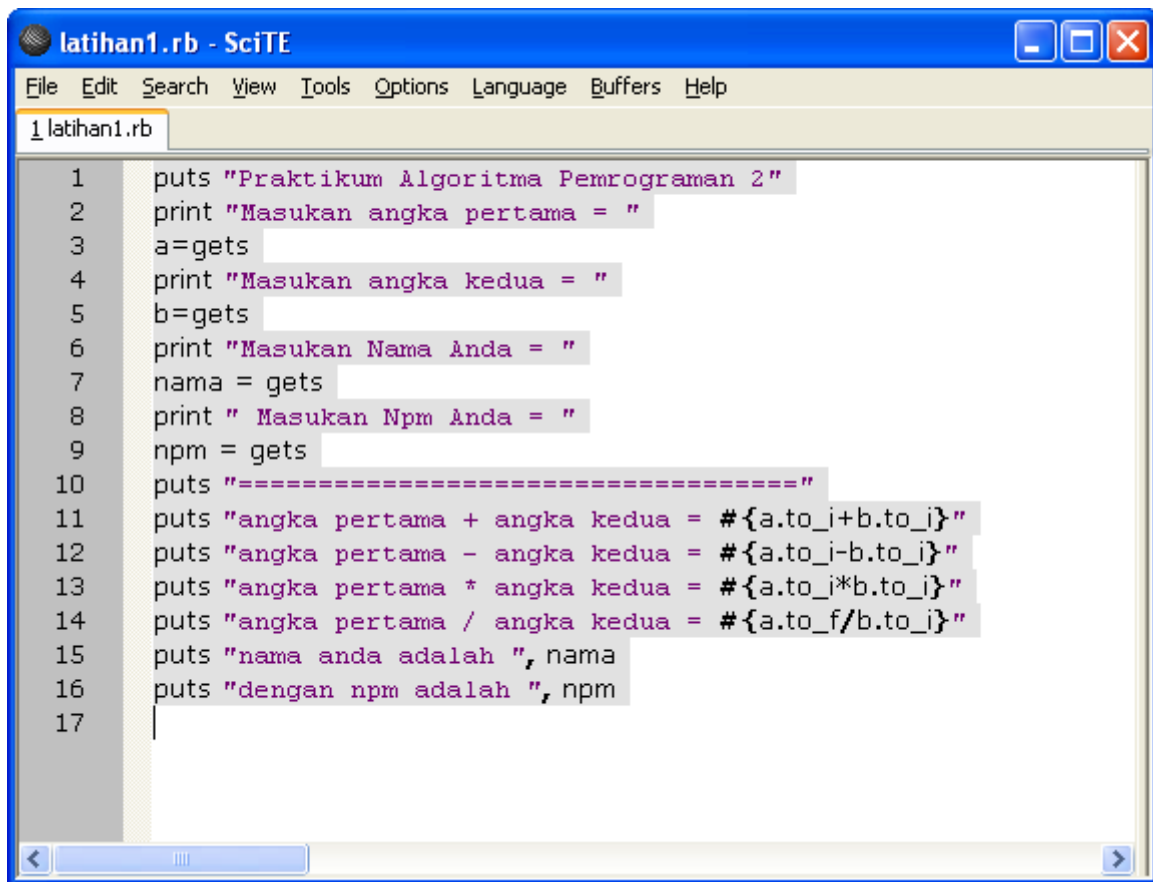


```
C:\WINDOWS\system32\cmd.exe

C:\>cd Ruby
C:\Ruby>cd bin
C:\Ruby\bin>ruby latihan1.rb
Praktikum Algoritma Pemrograman 2
Masukan angka pertama = 25
Masukan angka kedua = 10
Masukan Nama Anda = Nuraini Purwandari
Masukan Npm Anda = 50406536
=====
angka pertama + angka kedua = 35
angka pertama - angka kedua = 15
angka pertama * angka kedua = 250
angka pertama / angka kedua = 2.5
nama anda adalah
Nuraini Purwandari
dengan npm adalah
50406536
C:\Ruby\bin>_
```



## KOREKSI LATIHAN PROGRAM



```
1 puts "Praktikum Algoritma Pemrograman 2"
2 print "Masukan angka pertama = "
3 a=gets
4 print "Masukan angka kedua = "
5 b=gets
6 print "Masukan Nama Anda = "
7 nama = gets
8 print " Masukan Npm Anda = "
9 npm = gets
10 puts "====="
11 puts "angka pertama + angka kedua = #{a.to_i+b.to_i}"
12 puts "angka pertama - angka kedua = #{a.to_i-b.to_i}"
13 puts "angka pertama * angka kedua = #{a.to_i*b.to_i}"
14 puts "angka pertama / angka kedua = #{a.to_f/b.to_i}"
15 puts "nama anda adalah ", nama
16 puts "dengan npm adalah ", npm
17 |
```

### P5.4 DaftarPustaka

- <http://www.ruby-lang.org/id/>
- <http://rubyforge.org/>
- <http://poignantguide.net/ruby/>
- <http://tryruby.hobix.com>