

A stylized, colorful illustration of a landscape. The foreground features rolling green hills with dark brown soil patches. On the left, there are two trees: one with green foliage and one with purple foliage. A small red bird is flying in the sky above the trees. The background consists of light blue and white wavy lines representing a sky or distant hills.

Introduction to Haskell

Tim Pengajar
IF1210 Dasar Pemrograman
Update: 12 Februari 2014

What is Haskell?

- Haskell adalah bahasa pemrograman (paradigma) fungsional
- Dibuat oleh Peyton Jones dan Hughes pada tahun 1998
- Nama Haskell diambil dari nama **Haskell B. Curry**
 - Pionir λ calculus (lambda calculus); teori matematika ttg fungsi dan telah menjadi inspirasi dalam perancangan sejumlah bahasa pemrograman fungsional

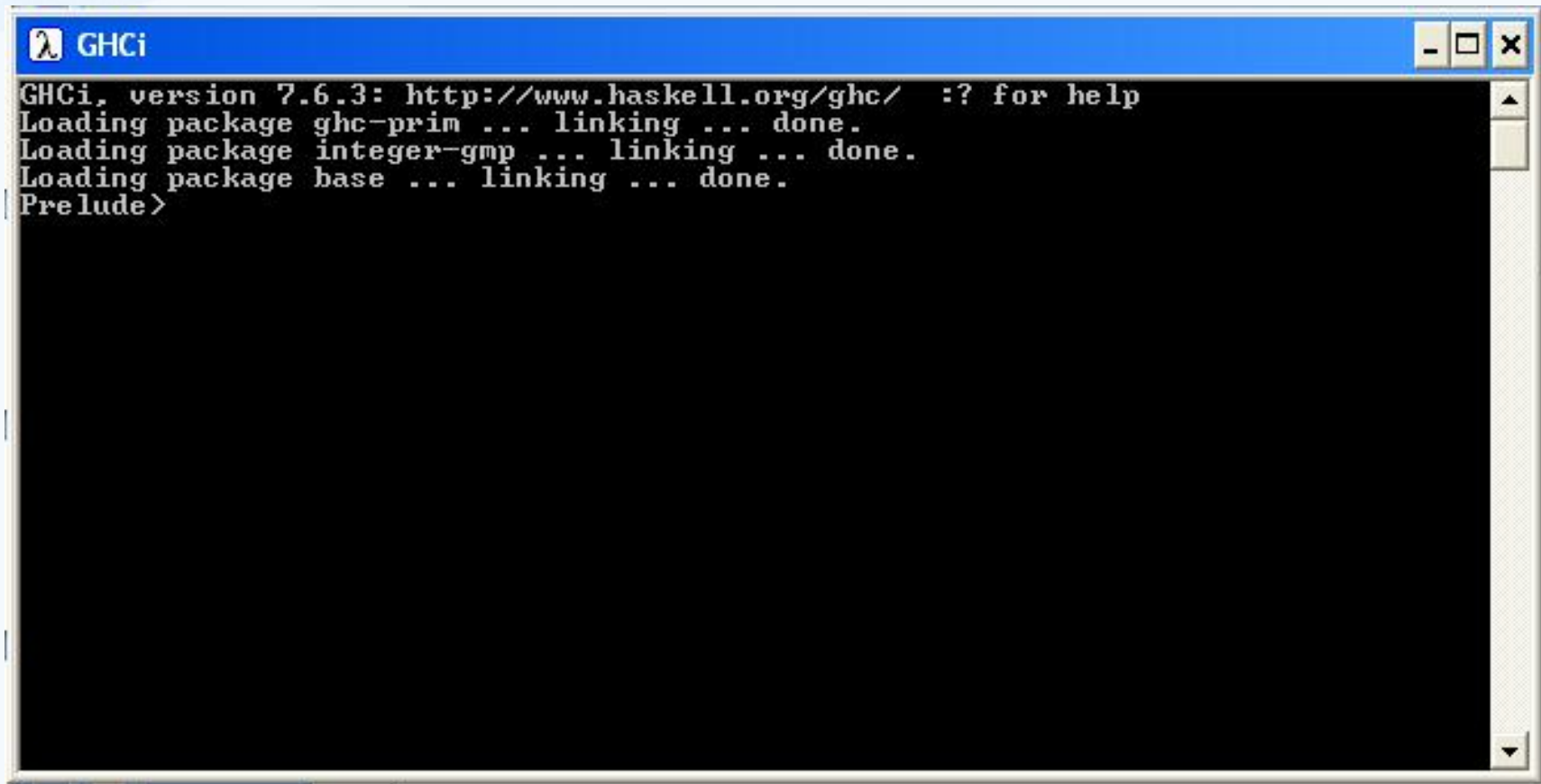
Features of Haskell

- *Concise programs*
- *Powerful type system*
- *List comprehensions*
- *Recursive functions*
- *Higher-order functions*
- *Monadic effects*
- *Lazy evaluation*
- *Reasoning about programs*

The Glasgow Haskell Compiler (GHC)

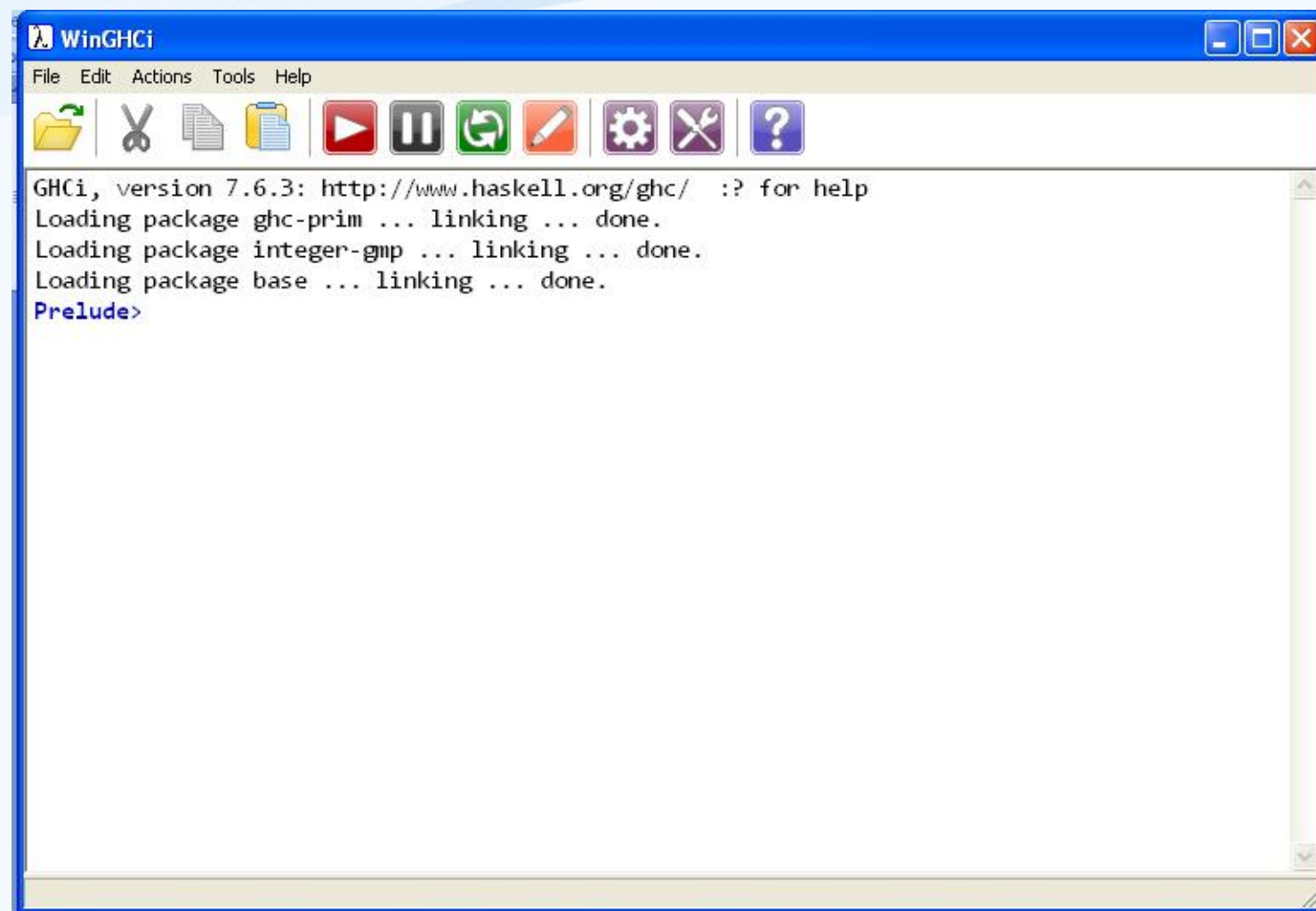
- Kompilator untuk Bahasa Pemrograman Haskell
- GHC memiliki dua komponen:
 - Interpreter interaktif (disebut GHCi)
 - Kompilator GHC
- Copyright 2002 - 2007, The University Court of the University of Glasgow

GHCi



```
GHCi, version 7.6.3: http://www.haskell.org/ghc/ :? for help
Loading package ghc-prim ... linking ... done.
Loading package integer-gmp ... linking ... done.
Loading package base ... linking ... done.
Prelude>
```

WinGHCi



Ekspresi

- Ekspresi : gabungan operan dan operator
- Operator adalah “sesuatu” paling dasar untuk mengoperasikan suatu nilai bertype tertentu
 - Contoh dalam fungsi pangkat3= $x*x*x$, operatornya adalah “*”
- Operan adalah “sesuatu” yang akan dioperasikan dengan operator tertentu
 - Operan dapat berupa suatu nilai yang bertype sesuai operator, atau hasil aplikasi fungsi
- Ekspresi fungsional :
 - Ekspresi aritmatika, logika
 - Ekspresi kondisional
 - Ekspresi rekursif

Contoh Ekspresi Aritmatika

Prelude> $3 + 4$

7

Prelude> $18 * 100$

1800

Prelude> $10/2$

5.0

Prelude> 8^2

64

Contoh Ekspresi Logika

Prelude> 8 > 13
False

Prelude> 8 < 15
True

Prelude> 15 >= 3
True

Prelude> 15 /= 15
False

Prelude> 15 == 15
True

Prelude> (15 < 10) || True
True

Prelude> (15 < 10) &&
True
False

Prelude> True && True
True

Prelude> True && False
False

Prelude> False || True
True

Contoh ekspresi kondisional

Prelude> if True then 5 else 3

5

Prelude> if 5 > 3 then 5 else 3

5

Prelude> if True then 5 else (if False then 3 else 5)

5

Memberi nama ekspresi dengan LET

```
Prelude> let a = 5 + 4
```

```
Prelude> a
```

9

```
Prelude> let pi = 3.14
```

```
Prelude> pi
```

3.14

```
Prelude> let benar = 5 > 3
```

```
Prelude> benar
```

True

Komentar

Prelude> -- ini komentar

Prelude> 3 + 4 -- ekspresi aritmatika
7

Tipe

- Konsep: $v :: T$ \rightarrow nilai v mempunyai tipe T

- Cek tipe:

```
Prelude> :type 4
```

```
4 :: Num a => a
```

```
Prelude> let x = 100
```

```
Prelude> :type x
```

```
x :: Integer
```

```
Prelude> let y = 1.5
```

```
Prelude> :type y
```

```
y :: Double
```

```
Prelude> :type True
```

```
True :: Bool
```

```
Prelude> :type 'a'
```

```
'a' :: Char
```

```
Prelude> :type "bandung"
```

```
"bandung" :: [Char]
```


Tipe Dasar

- *Bool* - logical values
- *Char* - single characters
- *String* - strings of characters
- *Int* - fixed-precision integers
- *Integer* - arbitrary-precision integers
- *Float* - single-precision floating-point numbers

Tipe List

- List adalah kumpulan nilai dengan tipe yang sama
- Contoh list:
 - [1,2,3,4,5]
 - [True, False, True]

Contoh manipulasi list

```
Prelude> head [1,2,3,4,5]  
1
```

```
Prelude> tail [1,2,3,4,5]  
[2,3,4,5]
```

```
Prelude> [1,2,3,4,5] !! 2  
3
```

```
Prelude> take 3 [1,2,3,4,5]  
[1,2,3]
```

```
Prelude> drop 3 [1,2,3,4,5]  
[4,5]
```

```
Prelude> length [1,2,3,4,5]  
5
```

```
Prelude> product [1,2,3,4,5]  
120
```

```
Prelude> sum [1,2,3,4,5]  
15
```

```
Prelude> reverse [1,2,3,4,5]  
[5,4,3,2,1]
```

```
Prelude> [1,2,3,4,5] ++ [6,7,8,9,10]  
[1,2,3,4,5,6,7,8,9,10]
```

```
Prelude> head []  
*** Exception: Prelude.head: empty list
```

Fungsi

- Fungsi akan memetakan satu atau beberapa nilai (domain) menjadi nilai yang lain (range)
- Jika tipe domain dan range tidak didefinisikan maka operator yang digunakan pada fungsi harus operator yang sesuai dengan nilai parameternya

- Contoh fungsi:

```
Prelude> let tambah2 x = x + 2
```

```
Prelude> tambah2 5
```

```
7
```

```
Prelude> let tambah x y = x + y
```

```
Prelude> tambah 2 5
```

```
7
```

```
Prelude> tambah 2.4 5.1
```

```
7.5
```

Penamaan Fungsi dan Parameter-nya

- Diawali dengan huruf kecil, boleh diikuti oleh huruf lain (besar atau kecil), angka, underscore (_), atau single quote (')
- Tidak boleh menggunakan keyword sebagai nama
*case class data default deriving do else
if import in infix infixl infixr instance
let module newtype of then type where*
- Nama fungsi dan parameter yang valid:
tambah5 x → nama fungsi: tambah5, parameter: x
rata2 daftar → nama fungsi: rata2, parameter: daftar
ambil_2 daftar → nama fungsi: ambil_2, parameter:
daftar

Script Haskell

contoh.hs berisi:

```
-- contoh 1  
tambah2 x = x + 2
```

Prelude> :load contoh.hs

[1 of 1] Compiling Main

(contoh.hs, interpreted)

Ok, modules loaded: Main.

**Main> tambah2 4*

6

**Main> take (tambah2 2) [1,2,3,4,5]*

[1,2,3,4]

Script Haskell (2)

Pada contoh.hs tambahkan:

```
rata2 daftar = div (sum daftar) (length daftar)
```

```
*Main>:reload
```

```
[1 of 1] Compiling Main  
interpreted )
```

```
( contoh.hs,
```

```
Ok, modules loaded: Main.
```

```
*Main> rata2 [1,2,3,4]
```

```
2
```

Command pada GHCi/WinGHCi

:load name	load script name
:reload	reload current script
:edit name	edit script name
:edit	edit current script
:type expr	show type of expr
:?	show all commands

Aturan Indentasi

- Indentasi menjadi hal yang penting
- Salah indentasi akan mengakibatkan kesalahan program

Aturan Indentasi (2)

Cobalah ketiga contoh di bawah ini, mana yang error:

```
tambah2 x = x + 2
```

```
tambah2 x =  
    x + 2
```

```
tambah2 x =  
x + 2
```


Referensi

- *Programming in Haskell*; Graham Hutton; University of Nottingham, Draft of August 22, 2003
- *Haskell The Craft of Functional Programming*; Simon Thompson; Second Edition; Addison-Wesley