FUNCTIONAL PROGRAMMING NO.1 INTRODUCTION TO HASKELL

Tatsuya Hagino
hagino@sfc.keio.ac.jp

lecture slide URL

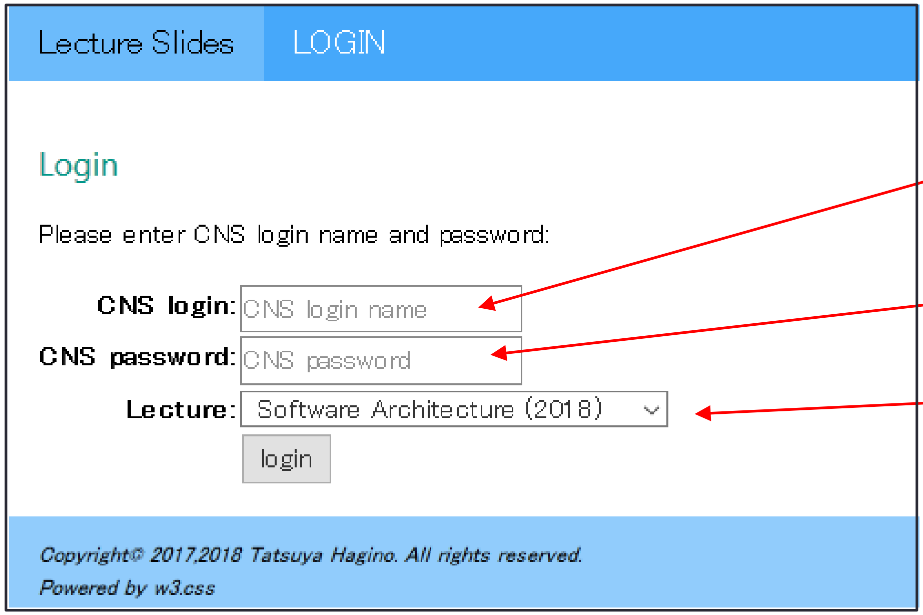
https://vu5.sfc.keio.ac.jp/slide/

Lecture Slide System

- Please access to:

https://vu5.sfc.keio.ac.jp/slide/

- Select: Fundamentals of Logic



Login

Please enter CNS login name and password:

CNS login:	CNS login name
CNS password:	CNS password
Lecture:	Software Architecture (2018)

Copyright© 2017,2018 Tatsuya Hagino. All rights reserved.
Powered by w3.css

Functional Programming

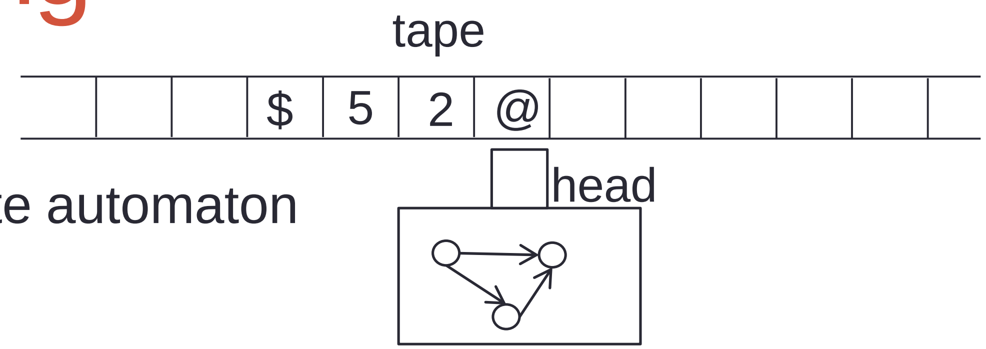
- Procedural Programming
  - write steps to solve the problem
  - programs are executed line by line
  - most popular way of programming
  - ex. FORTRAN, C, Java, Javascript, …

- Logic Programming
  - write logical formulae (or rules) to solve the problem
  - the order of rules does not matter
  - no side effect
  - no assignment
  - ex. PROLOG

- Functional Programming
  - combine functions to solve the problem
  - order of evaluation does not matter
  - no side effect
  - no assignment
  - ex. LISP, FP, ML, Haskell

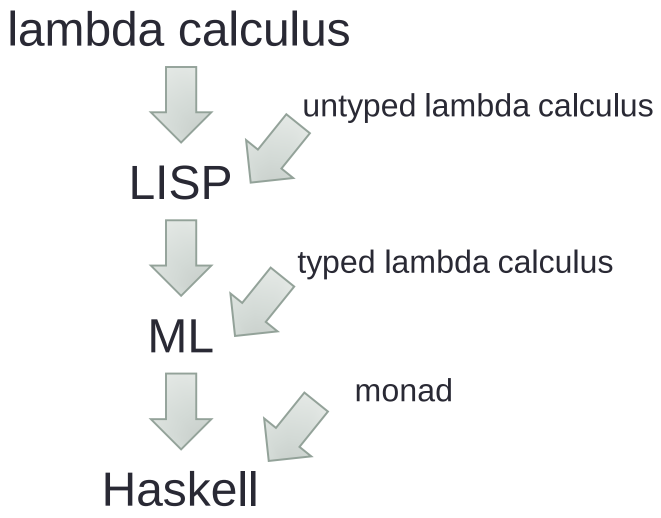
Model of Computing

- Turing Machine
  - an infinite tape and a finite state automaton
  - Universal Turing Machine



- Recursive Function
  - primitive recursive function + mu operator
  - f(x+1) = (x+1) × f(x), f(0) = 1

- Lambda Calculus
  - function abstraction + function application
  - (λx.(λy.xy))(λx.x) → (λy.(λx.x)y) → λy.y



Programming Language Haskell

- Pure functional programming language
  - no side effect
  - referential transparency
- Strong typing
  - type checked before compilation
- Polymorphism
  - Functions may be applied to multiple type values.
- Non-strict
  - lazy evaluation
- Monad
  - order evaluation

Haskell Brooks Curry

- American mathematician and logician (1900/9/12 - 1982/9/1)
- combinatory logic
  - S, K, I
  - equivalent to lambda calculus
- Curry's Paradox
  - If this sentence is true, then Japan is in Europe.
- Curry-Howard correspondence
  - logic ⇔ computation
  - proof as program
- Currying
  - $(A \times B \to C) \to (A \to (B \to C))$

Installing Haskell to Mac OS X

- Start Terminal application (in /Applications/Utilities folder)
  - Use Finder or LaunchPad

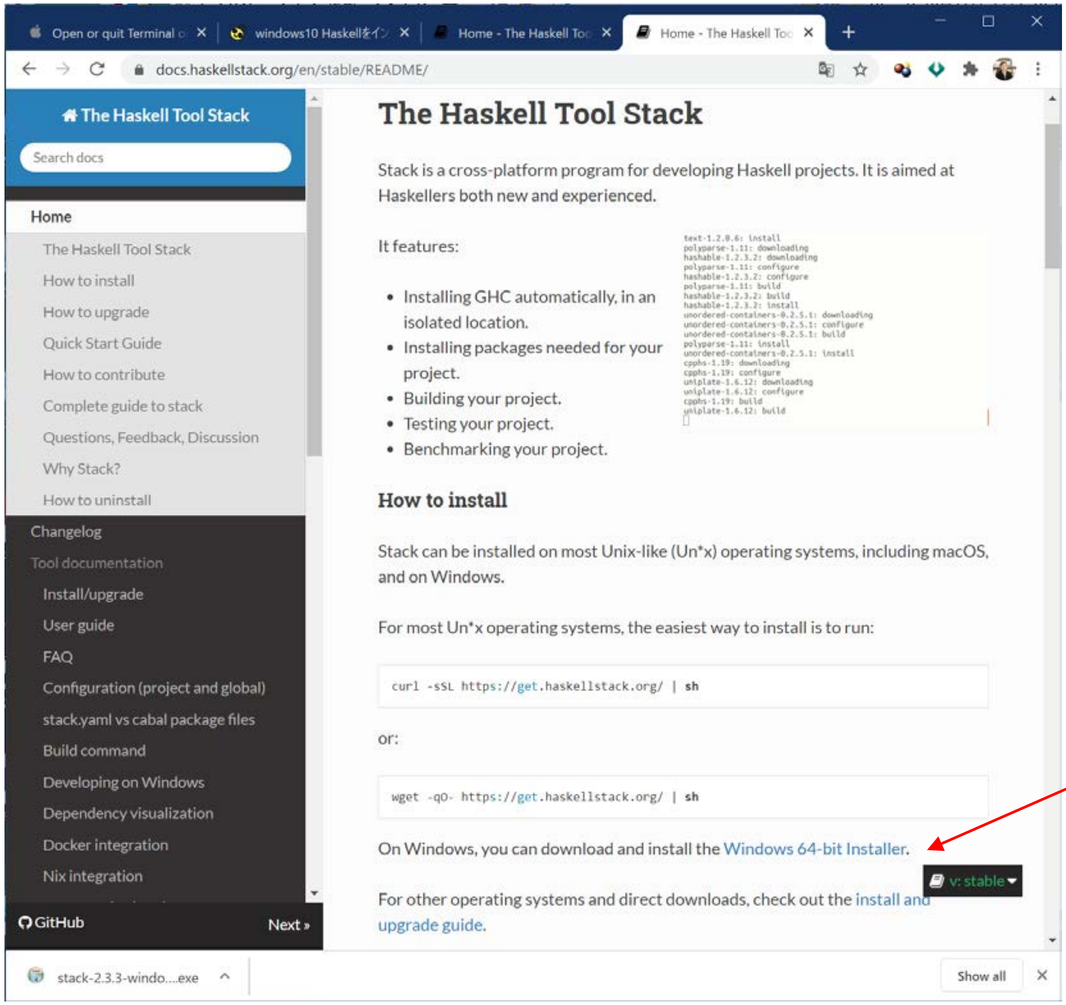
```
% brew install stack
  (message output)

% rehash

% stack setup
  (message output)

% stack path
  (message output)

% stack ghci
GHCi, version 8.8.4: http://www.Haskell.org/ghc/  :? for help
Prelude> :quit
Leaving GHCi
%
```

If necessary (stack command not found)

Test the installation

Installing Haskell to Windows (1)

- Google 'Haskell Tool Stack'
  - https://docs.haskellstack.org/en/stable/README/



The Haskell Tool Stack

Stack is a cross-platform program for developing Haskell projects. It is aimed at Haskellers both new and experienced.

It features:

- Installing GHC automatically, in an isolated location.
- Installing packages needed for your project.
- Building your project.
- Testing your project.
- Benchmarking your project.

```
text-1.2.0.6: install
polyparse-1.11: downloading
hashable-1.2.3.2: downloading
polyparse-1.11: configure
hashable-1.2.3.2: configure
polyparse-1.11: build
hashable-1.2.3.2: build
hashable-1.2.3.2: install
unordered-containers-0.2.5.1: downloading
unordered-containers-0.2.5.1: configure
unordered-containers-0.2.5.1: build
polyparse-1.11: install
unordered-containers-0.2.5.1: install
cpphs-1.19: downloading
cpphs-1.19: configure
uniplate-1.6.12: downloading
uniplate-1.6.12: configure
cpphs-1.19: build
uniplate-1.6.12: build
```

How to install

Stack can be installed on most Unix-like (Un*x) operating systems, including macOS, and on Windows.

For most Un*x operating systems, the easiest way to install is to run:

```
curl -sSL https://get.haskellstack.org/ | sh
```

or:

```
wget -qO- https://get.haskellstack.org/ | sh
```

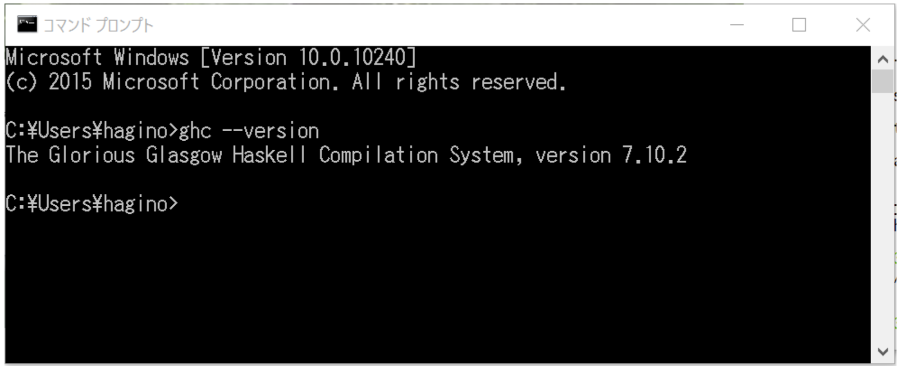
On Windows, you can download and install the Windows 64-bit Installer.

For other operating systems and direct downloads, check out the install and upgrade guide.

Download
Windows 64-bit Installer

Installing Haskell to Windows (2)

- Install the stack by using the installer
- Start Command Prompt



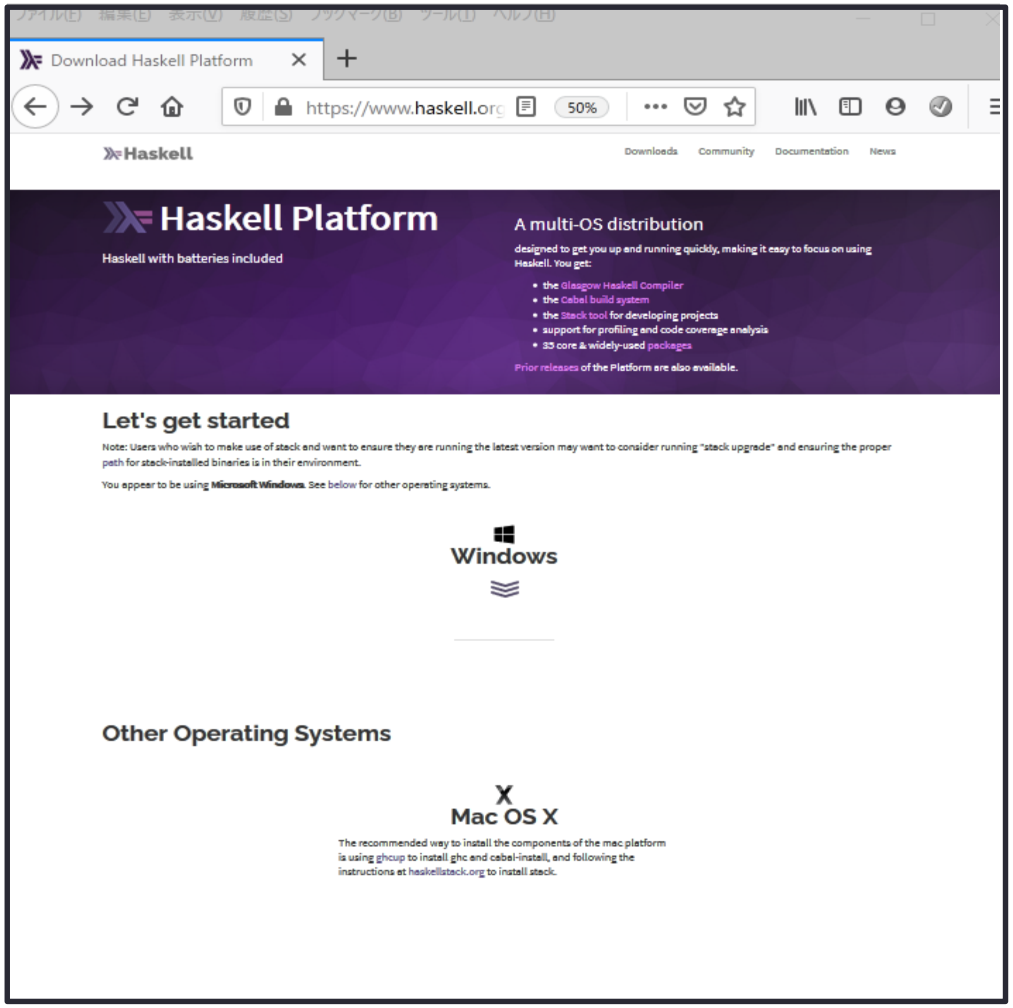
```
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:¥Users¥hagino>ghc --version
The Glorious Glasgow Haskell Compilation System, version 7.10.2

C:¥Users¥hagino>
```

```
C:Users¥hagino>stack setup
  (message output)

C:Users¥hagino>stack path
  (message output)

C:Users¥hagino>stack ghci
GHCi, version 8.8.4: http://www.Haskell.org/ghc/  :? for help
Prelude> :quit
Leaving GHCi
C:Users¥hagino>
```

Test the installation

Other OS

- https://haskell.org/platform/


ファイル(F) 編集(E) 表示(V) 履歴(S) ブックマーク(B) ツール(T) ヘルプ(H)
Download Haskell Platform
https://www.haskell.org
50%
Haskell
Downloads
Community
Documentation
News

Haskell Platform

Haskell with batteries included

A multi-OS distribution

designed to get you up and running quickly, making it easy to focus on using Haskell. You get:

- the Glasgow Haskell Compiler
- the Cabal build system
- the Stack tool for developing projects
- support for profiling and code coverage analysis
- 35 core & widely-used packages

Prior releases of the Platform are also available.

Let's get started

Note: Users who wish to make use of stack and want to ensure they are running the latest version may want to consider running "stack upgrade" and ensuring the proper path for stack-installed binaries is in their environment.

You appear to be using **Microsoft Windows**. See below for other operating systems.

Windows

Other Operating Systems

Mac OS X

The recommended way to install the components of the mac platform is using ghcup to install ghc and cabal-install, and following the instructions at haskellstack.org to install stack.

Hello, World!

- Write the first Haskell program.
  1. Write the following line as "hello.hs".

```
main = putStrLn "Hello, World!"
```

  2. Compile it using "stack ghc" command.

```
% stack ghc hello.hs
[1 of 1] Compiling Main ( hello.hs, hello.o )
Linking hello ...
```

  3. Execute the compiled program.

```
% hello
Hello, World!
```

Windows

```
C:¥> hello.exe
Hello, World!
```

Direct/Interactive Execution

- Direct execution
- Execute programs without compiling
- use "stack runghc" command

```
% stack runghc hello.hs
Hello, World!
```

- Use interactively by "stack ghci" command.

```
% stack ghci
GHCi, version 8.10.1: http://www.haskell.org/ghc/  :? for help
Prelude> 1 + 2
3
Prelude> putStrLn "Hello, World!"
Hello, World!
Prelude> :quit
```

main action

```
main = putStrLn "Hello, World!"
```

- This is the definition of variable "main".
- The value of "main" is not a function, but an action.
- "putStrLn" is a function.
  - "putStrLn" takes a string literal "Hello, World!".
  - "putStrLn" returns an action which outputs the given string.
- When a Haskell program is executed, the main action is evaluated (i.e. executed).
- Function application may not need any parenthesis.

```
putStrLn "Hello, World!"
(putStrLn "Hello, World!")
putStrLn("Hello, World!")
```

- The same meaning: "Hello, World!" applied to "putStrLn" function

Combining two actions

- Write "Hello, World!" and "Hello, SFC!" in two lines.

```
main = putStrLn "Hello, World!" "Hello, SFC!"
```

⇩ means

```
main = (putStrLn "Hello, World!") "Hello, SFC!"
```

- does not work

```
main = putStrLn "Hello, World!"
       putStrLn "Hello, SFC!"
```

- does not work

```
main = (putStrLn "Hello, World!") >> (putStrLn "Hello, SFC!")
```

- does work

```
main = do putStrLn "Hello, World!"
          putStrLn "Hello, SFC!"
```

- does work

do syntax

- Input a name and output a message.

```
main = do putStrLn "What is your name?"
          name <- getLine
          putStrLn ("Hello, " ++ name ++ "!")
```

Align actions you want to combine

- **`getLine`**
  - an action for getting a line from the standard input (i.e. terminal).
- **`name <- getLine`**
  - not assignment statement
  - If the action (**`getLine`**) is successful, the value is bound to **`name`**
- `++`
  - binary operator
  - concatenate two strings

Exercise 1-1

- Write a Haskell program friend.hs to ask two names and output a message saying they are friends.

```
% stack runghc friend.hs
Input a name?
Taro
Input another name?
Hanako
Taro and Hanako are friends.
```

user input

- Please submit only a .hs file (not executable one)
- Deadline of submitting homework is Saturday of the same week.
- This course will be evaluated by submission of exercises.
  - Attendance is of course important by default.