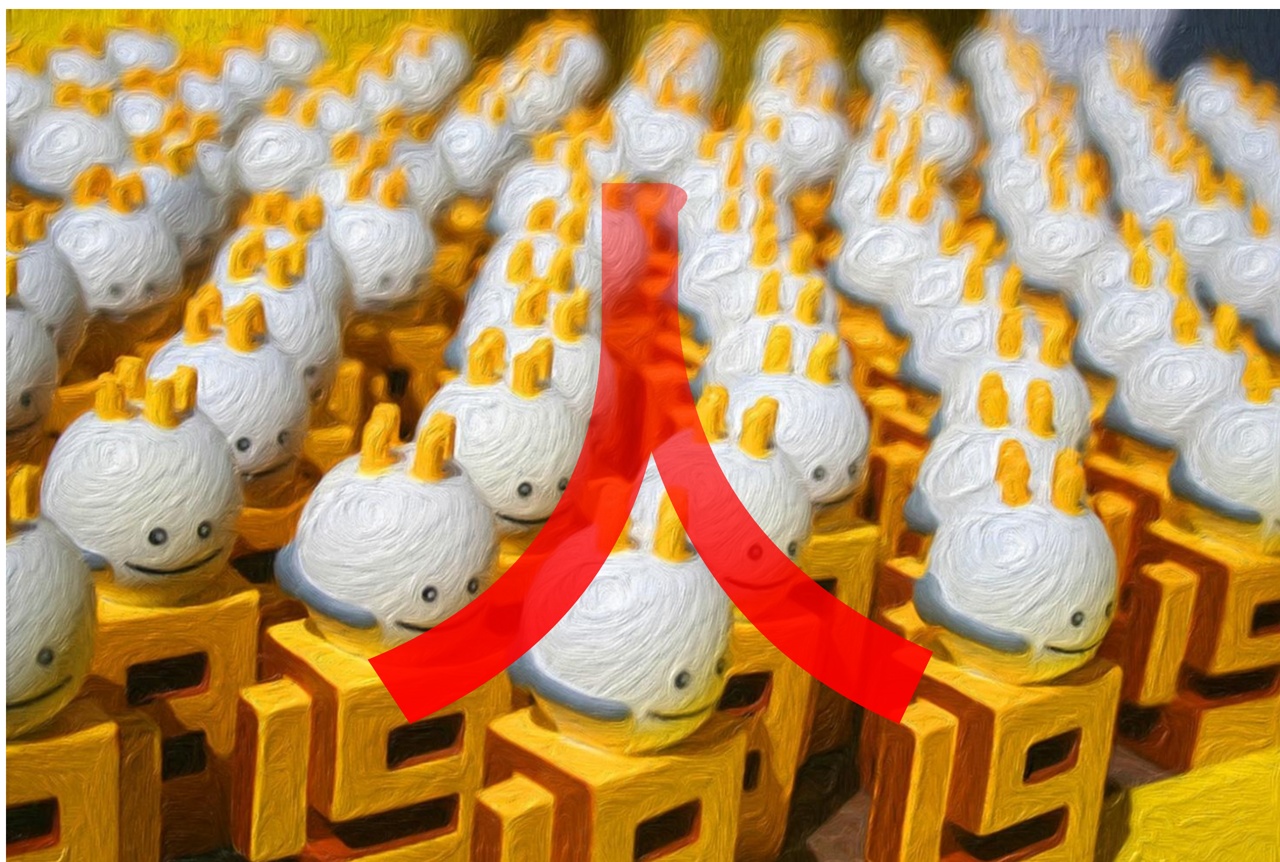


Quick Reference for the Channel 9 Lecture Series on Functional Programming by Dr. Erik Meijer.

This reference is intended to be used as an initial reference for the lecture series. It only covers the absolute basics – the essence of the first two lectures.

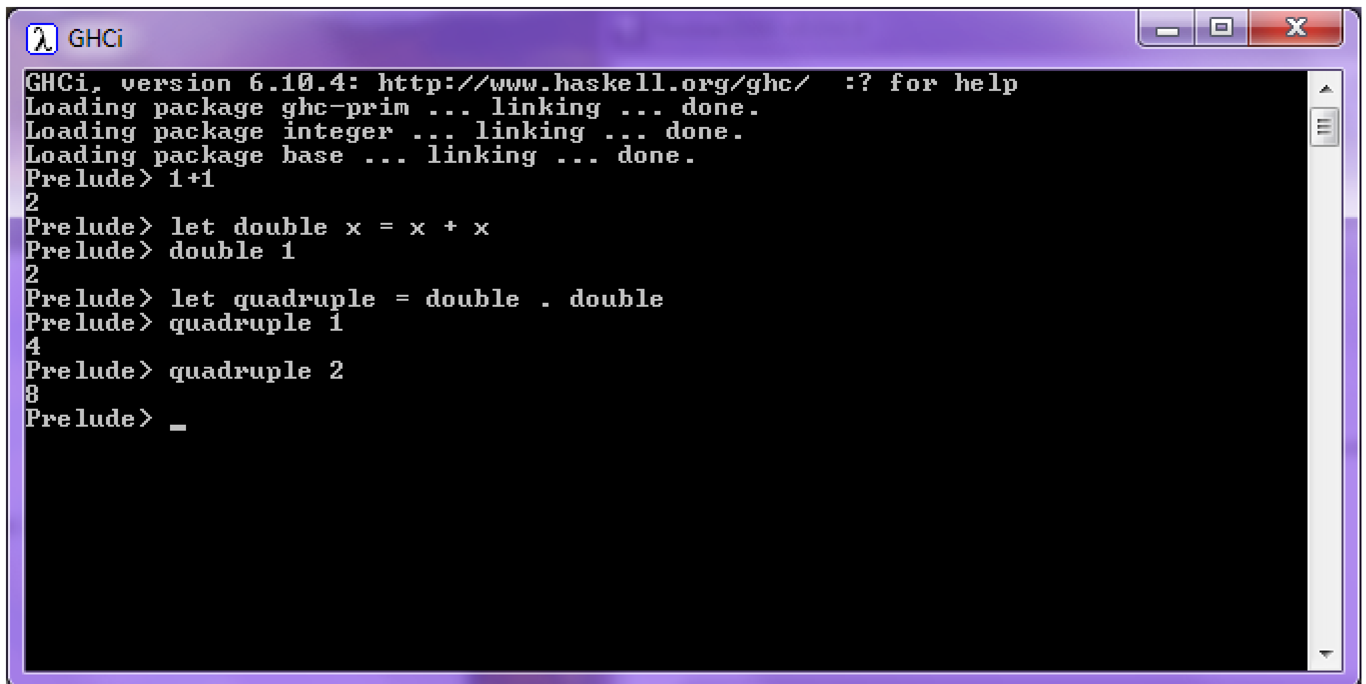
To use multi-line function definitions, write them in a text file and load them into the interactive promt.

(λx → succ $ 1 – 1) ()

1 Read, Eval, Print – Loop (REPL)

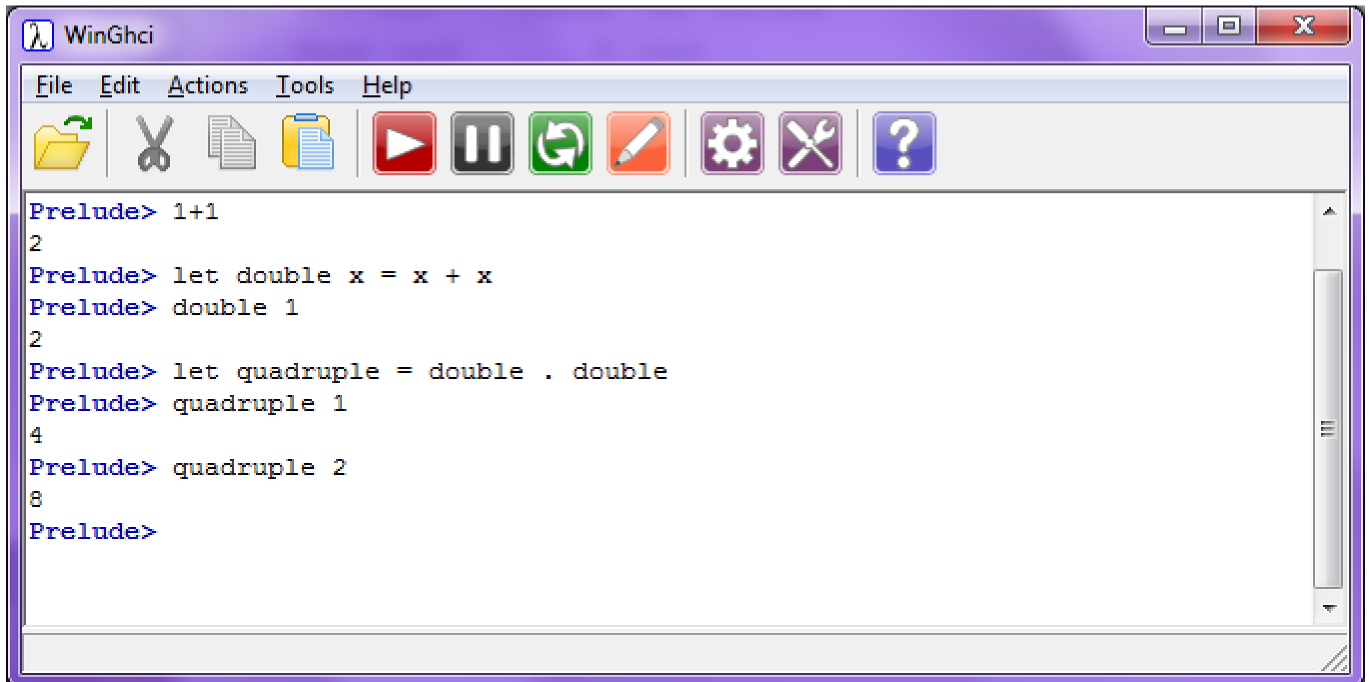
1.1 REPL # 1 GHCi

This is the Glasgow Haskell Compiler Interactive (GHCi) promt. Write an expression and press enter. Then the value of the expression will be written next.



1.1 REPL #2 WinGHCi

This essentially the same as the GHCi REPL, but it is faster and lighter on the eyes.



2 Concepts

2.1 Application, Abstraction & Composition

The REPL below shows three crucial concepts of functional programming: (**i**) ***function application***, (**ii**) ***function abstraction*** and (**iii**) ***function composition***

WinGhci

File Edit Actions Tools Help

```
Prelude> 1+1
2
Prelude> let double x = x + x
Prelude> double 1
2
Prelude> let quadruple = double . double
Prelude> quadruple 1
4
Prelude> quadruple 2
8
Prelude>
```

application

abstraction

composition

1 + 1 is infix application of the **+** function to **1** and **1** (used for “operator functions” such as **+**)

+ 1 1 is the prefix application of the **+** function to **1** and **1**

let double x = x + x is the *abstraction* of **x + x** over **double**

double . double is the *composition* of **double** and **double** where “**.**” is the infix *composition* operator

1 + 1 demonstrates ***application***

let double x = x + x demonstrates ***abstraction*** and ***application***

let quadruple = double . double demonstrations ***composition***, ***abstraction*** and ***application***

This function

```
let quadruple = double . double
```

may be rewritten as

```
let quadruple = \x → double (double x)
```

where

```
\x → double (double x)
```

corresponds to the C# lambda expression

```
x => double(double(x))
```

(λx → succ $ 5 – 1) ()