Haskell Lecture 1

Introduction

Hello World?

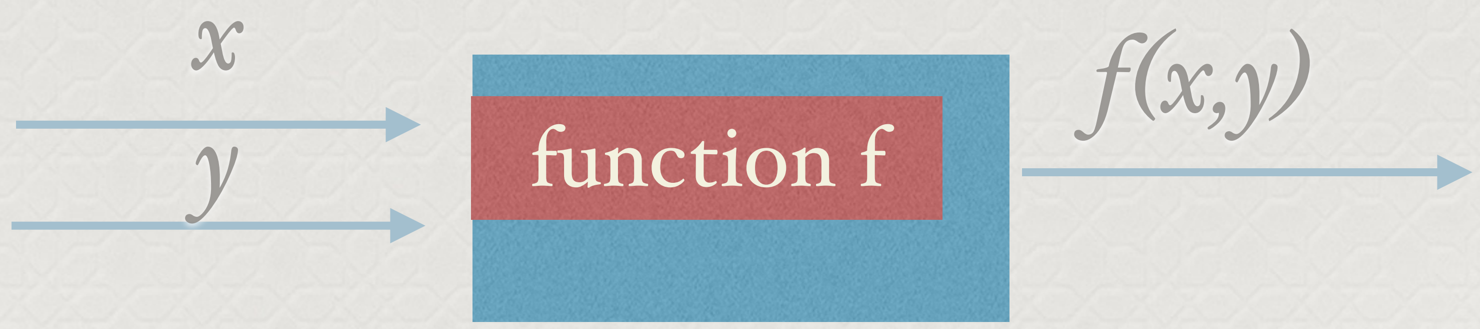
- *Example program to read and write*
- *main is the entry point into program*
- *do is used for sequencing*
- *getLine and putStrLn are input output functions*

Haskell Philosophy

- *A program is a collection of functions*
    - *Function always returns same value for given inputs*
    - *Variables are immutable, as math variables*
- *Pure functions follow above principles*
- *Pure functions do not do I/O, or have side effects*
- *Programs must communicate with world - impure functions*
- *Pure functions can be automatically run in a thread*

Functions

- *A function takes one or more arguments and produces one output. These are functions in the math sense, we always get the same output (and nothing else happens) given particular values of inputs.*

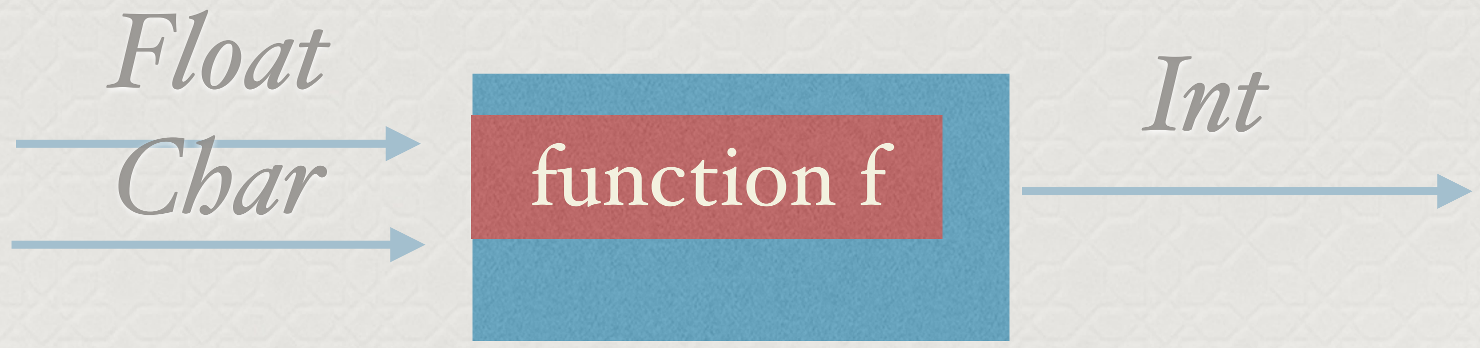


Types

- *Collections of values, such as*
  - *Int*
  - *Bool*
  - *Float*
  - *Integer*

Functions

- *The inputs and arguments to a function are described by their types*



Definitions

- *Associate a name with a particular type, syntax*

```
name :: type
name = expression
```

- *Example*

```
bakersDozen :: Int
bakersDozen = 12 + 1
```

- *note that name starts with lower case and type starts with upper case - this is a language rule*

Function Definitions

- *syntax of declaration of function type*

  name :: t1 -> t2 -> ... -> tk -> t

- *semantics*

  name is function name

  t1 through tk are formal parameter types

  t is the return type

- *After the declaration of type we write the function definition*

Function Definition Example

- *Increment - to add 1 to a value*

```
increment :: Int -> Int
increment n = n + 1
```

- *Note that the parameter is not enclosed in parentheses*

GHC and GHCI

- *Glasgow Haskell Compiler Interactive*
- *Use this to test functions interactively without having to run a complete program (compiled using GHC)*
- *Show examples bakersDozen, increment, add and multiply*
- *Show how to use :type, :info, :load, :reload, :quit*

Error Messages

- *At the start, these will be nearly meaningless to a new Haskell programmer*
- *Pay attention to where the error occurred, that is useful*

Types - Bool

- *boolean values are either true or false; in Haskell the type is Bool, and the possible values are True and False*
- *Operators for Bool are:*

&&.	*and*
||.	*or*
not.	*not*

- *Define functions for exclusive or and nand*

Integer Types

- *The whole numbers, including negative values and zero, are represented in an integer type*
- *Int has a maximum and minimum value, the largest (and smallest) values which can be represented by bits fitting in a register*
- *Integer has unlimited (theoretically) size*
- *Operators include +, -, ^, *, div, mod, abs, and negate*
- *Be careful using - to change a sign of a value to negative, Haskell gets confused at times between unary and binary minus*
- *Relational operators: as usual >, >=, <, <=, ==, but not equal is /=*

Real Types

- *Float and Double have same meaning in Haskell as float and double in Java*

Char and String Types

- *use Char to represent a single character, literal is defined the same way as in Java, for example 'a'*
- *Escape sequence is used in the same way as in Java, for example '\n' is newline*
- *String literals are also written the same way as in Java, "This is a string literal"*
- *Strings are actually lists of Char*
- *The String concatenation operator, ++, is actually the concatenation operator for lists.*