CS314. Функциональное программирование

Лекция 13. Использование инструментов и библиотек Haskell в разработке ПО

Направление «Фундаментальная информатика и информационные технологии»
Институт математики, механики и компьютерных наук имени И. И. Воровича
Южный федеральный университет

3 ноября 2017 г.

Содержание

Содержание

Основные инструменты

- Компилятор (ghc).
- Интерпретатор и отладчик (ghci).
- GHC runtime + средства профилирования (RTS).
- HPC — Haskell Code Coverage.
- Haddock.
- Генераторы лексических анализаторов Alex и парсеров Happy.

Внешние зависимости и управление сборкой

- Haskell Platform
- Коллекция пакетов Hackage и cabal-install

```
$ cabal update
$ cabal list
$ cabal install primes
```

- Stackage и LTS Haskell, утилита stack

```
$ stack new my-project
$ cd my-project
$ stack setup
$ stack build
$ stack ghci
$ stack test
$ stack exec my-project-exe
```

```
LICENSE
Setup.hs
app
  Main.hs
my-project.cabal
src
  Lib.hs
stack.yaml
test
  Spec.hs
```

Важнейшие прикладные библиотеки

- Быстрые структуры данных: vector, containers, unordered-containers, hashtables.
- Математические вычисления: hmatrix.
- Регулярные выражения: regex-pcre.
- Пользовательский интерфейс: GTK, wxhaskell.
- Базы данных: HDBC, HaskellDB, Persistent.
- Парсеры: Parsec, attoparsec, aeson, cassava.
- Сетевое взаимодействие: Network, blaze-builder/cereal.
- Веб-фреймворки: Yesod, Snap, Servant.
- FFI — Foreign Function Interface.

Библиотеки для разработки

- Тестирование: HUnit, QuickCheck.
- Профилирование: criterion.
- Логирование: hslogger.
- Конфигурирование: configurator.
- Параметры командной строки: optparse-applicative

Содержание

Биржевые котировки

```
day,close,volume,open,high,low
2017/10/11,156.5500,16861450.0000,155.9700,156.9800,155.7500
2017/10/10,155.9000,15603520.0000,156.0550,158.0000,155.1000
2017/10/09,155.8400,16243080.0000,155.8100,156.7300,155.4850
2017/10/06,155.3000,17223790.0000,154.9700,155.4900,154.5600
2017/10/05,155.3900,21215870.0000,154.1800,155.4400,154.0500
...
```

- CSV-файл
- Даты
- Валюты

Задача

```
Usage: stockdata (-f|--fname ARG) [-d|--days] STATISTIC FIELD
  Print statistics on stock quotes data

Available options:
  -f,--fname ARG      File with quotes data
  -d,--days           Print number of days between min/max
  -h,--help           Show this help text

Available statistics:
  mean                Compute mean
  min                 Compute minimum
  max                 Compute maximum

Available fields:
  open                open
  ...
```

Пример

```
$ stockdata --fname ../quotes.csv -d mean close
Days between min/max close:
291
Statistics: mean on close
138.1830
$ stockdata --fname ../quotes.csv max volume
Statistics: maximum on volume
111837300.0000
```

Основные подзадачи и соответствующие библиотеки

- Чтение csv-файла: cassava
- Работа с датами: time
- Работа с валютой (данные с фиксированной точкой): numbers
- Разбор параметров командной строки: optparse-applicative
- Вычисление статистик: vector

Числа с фиксированной точкой (numbers)

```
import Data.Number.Fixed

type Eps4 = EpsDiv10 (EpsDiv10 (EpsDiv10 (EpsDiv10 Eps1)))
type Fixed4 = Fixed Eps4
```

```
ghci> let a = 123.2387 :: Fixed4
ghci> a+a
246.4774
ghci> a * 10
1232.3871
ghci> a+a+a+a+a+a+a+a+a+a
1232.3872
ghci> a+a
246.4774
ghci> a+a+a+a-a-a
246.4775
```

Тип для сведений о котировках

```
import Data.Time.Calendar
import Data.Time.Format
import GHC.Generics (Generic)

data QuoteData = QD {
                    day :: Day,
                    close :: Fixed4,
                    volume :: Fixed4,
                    open :: Fixed4,
                    high :: Fixed4,
                    low :: Fixed4
                   }
                   deriving (Generic, Show)
```

Подготовка к чтению CSV-файла (cassava, vector, time)

- CSV-файл $\Longrightarrow$ `Vector QuoteData`

```
import Data.Vector
import Data.Csv
import Data.ByteString.Char8 (unpack)

instance FromField Fixed4 where
  parseField s = return (read $ unpack s)

instance FromField Day where
  parseField s = parseTimeM False defaultTimeLocale
                            "%Y/%m/%d" (unpack s)

instance FromNamedRecord QuoteData
instance DefaultOrdered QuoteData
```

Чтение CSV-файла

```
do
  csvData <- BL.readFile fname
  case decodeByName csvData of
    Left err -> putStrLn err
    Right (_, quotes) -> do
      ...
```

- `fname` — имя CSV-файла
- `csvData` — `Bytestring`
- `quotes` — `Vector QuoteData`

Вычисление статистической информации

```haskell
data Operation = Mean | Min | Max

instance Show Operation where
  show Mean = "mean"
  show Min = "minimum"
  show Max = "maximum"

op2fun Mean = mean
op2fun Min = minimum
op2fun Max = maximum

mean v = sum v / (fromIntegral $ length v)
```

```haskell
data QField = Op | Cl | High | Low | Vol

instance Show QField where
  show Op = "open"
  show Cl = "close"
  show Vol = "volume"
  show High = "high"
  show Low = "low"

f2fun :: QField -> QuoteData -> Fixed4
f2fun Op = open
f2fun Cl = close
f2fun Vol = volume
f2fun High = high
f2fun Low = low
```

```haskell
import qualified Data.Vector as V

computeStatistics :: Operation -> QField ->
                     V.Vector QuoteData -> Fixed4
computeStatistics op qf qv = op2fun op $ V.map (f2fun qf) qv

daysBetweenMinMax qf quotes =
    abs $ diffDays (day minQuote)
                   (day maxQuote)
  where
    minQuote = minimumBy (comparing (f2fun qf)) quotes
    maxQuote = maximumBy (comparing (f2fun qf)) quotes
```

Разбор параметров командной строки (optparse-applicative)

```
data Params = Params {
                fname :: String,
                daysBetween :: Bool,
                op :: Operation,
                qf :: QField
              }
```

```
params = Params
  <$> strOption (long "fname" <> short 'f' <>
                 help "File with quotes data")
  <*> switch (long "days" <> short 'd' <>
              help "Print number of days between min/max")
  <*> subparser (metavar "STATISTIC" <>
       commandGroup "Available statistics:" <>
       command "mean" (Mean `withInfo` "Compute mean") <>
       command "min" (Min `withInfo` "Compute minimum") <>
       command "max" (Max `withInfo` "Compute maximum"))
  <*> subparser (metavar "FIELD" <>
       commandGroup "Available fields:" <>
       command "open" (Op `withInfo` "open") <>
       command "close" (Cl `withInfo` "close") <>
       command "low" (Low `withInfo` "low") <>
       command "high" (High `withInfo` "high")  <>
       command "volume" (Vol `withInfo` "volume"))
    where withInfo op str = info (pure op) (progDesc str)
```

```haskell
work :: Params -> IO ()
work Params {..} = do
  csvData <- BL.readFile fname
  case decodeByName csvData of
    Left err -> putStrLn err
    Right (_, quotes) -> do
      when daysBetween $ do
        putStrLn $ "Days between min/max " ++ show qf ++ ":"
        print $ daysBetweenMinMax qf quotes
      putStrLn $ "Statistic: " ++ show op ++ " on "
                                          ++ show qf
      print $ computeStatistics op qf quotes
```

Функция main

```
main :: IO ()
main = execParser opts >>= work
 where
  opts = info (params <**> helper)
              (fullDesc <>
               progDesc "Print statistics on stock quotes")
```

Основные подзадачи и соответствующие библиотеки

- Чтение csv-файла: cassava
- Работа с датами: time
- Работа с валютой (данные с фиксированной точкой): numbers
- Разбор параметров командной строки: optparse-applicative
- Вычисление статистик: vector