Esai Refleksi Parser Combinator

Attribute	Information
Nama	Fahdii Ajmalal Fikrie
NPM	1906398370
URL Repositori	https://github.com/fahdikrie/parsing-with-haskell-parser-combinators

Refleksi Pembelajaran *Parser Combinator* Menggunakan Haskell pada Implementasi *Parsing Version Number* dan *Subtitle*

Untuk tugas UTS kali ini, saya memilih salah satu implementasi *parser combinator* dengan bahasa Haskell yakni proyek *parsing* Version Number yang dan *parsing subtitle* dengan format `.srt`. Sebagai atribusi, proyek ini dibuat oleh pengguna Github dengan *username* lettier.

Kemudian, hasil refleksi akan saya jabarkan dalam bentuk poin sebagai berikut:

1. **Penggunaan Library ReadP**

   Proyek ini dibuat salah satunya dengan bantuan ReadP yang sudah tersedia secara *out-of-the-box* pada library base (Haskell Basic Library) di instalasi Haskell. ReadP serupa dengan Parsec, yakni sama-sama berfungsi sebagai *parser* dari suatu teks menjadi ke dalam tipe data.

   Pada ReadP, saya dikenalkan dengan P *data type* yakni sebagai berikut:

   ```
   data P a = Get (Char -> P a)
     | Look (String -> P a)
     | Fail
     | Result a (P a)
     | Final [(a,String)]
     deriving Functor
   ```

   P *data type* berisi definisi dari operasi yang dapat dilakukan pada ReadS.

2. **Logika di Balik Implementasi *Version Number Parser***

   Program *parser version number* tujuannya adalah untuk mengekstrak versi number dari dependensi yang digunakan oleh Gifcurry (aplikasi pembuat GIF yang ditulis menggunakan Haskell). *Parser* ini terdiri atas empat fungsi utama, yaitu `parseVersionNumber`, `parseSeparator`, `parseNumber`, dan `parseNotNumber`.

Mula-mula terdapat pemanggilan fungsi `readFile` untuk membaca file input yang ingin kita *parse version number*-nya. Lalu dilakukan pemanggilan fungsi dari *library* ReadP, yaitu `ReadP_to_S`, diikuti dengan pemanggilan fungsi `parseVersionNumber` yang dikemas dalam sintaks *case notation*.

```
parseVersionNumber :: [String] -> ReadP [String]
parseVersionNumber nums = do
  _         <- parseNotNumber
  num       <- parseNumber
  let nums' = nums ++ [num]
  parseSeparator nums' parseVersionNumber
```

Pada fungsi `parseVersionNumber` di atas, terdapat penggunaan alternatif sintaks Monad, yakni *do notation*, yang melakukan pemanggilan fungsi `parseNotNumber` dan `parseNumber` untuk melakukan *parsing* karakter *number* dan *non-number* dari string hasil konversi dari `ReadP_to_S`.

Selanjutnya, hasil *parsing* angka disimpan dan kemudian dilakukan pemanggilan fungsi `parseSeparator` untuk menghilangkan separator berupa *punctiation* dari hasil *parsing number*, dan pemanggilan kembali fungsi `parseVersionNumber` secara rekursif.

3. **Hal Menarik pada Implementasi *Subtitle Parser***

   Dibanding implementasi *version number parser* sebelumnya, *subtitle parser* ini lebih kompleks untuk diimplementasikan. Mengingat objek yang di-*parsing—file subtitle* dengan format `.srt`—dinilai lebih kompleks pula.

   Satu yang menarik dari implementasi ini adalah pendeklarasian beberapa tipe data sesuai dengan yang terdapat pada format *file subtitle*, yakni `Tag`, `TaggedText`, `Timestamp`, `SrtSubtitleCoordinates`, dan `SrtSubtitle` (sebagai *main data type*).

   ```
   -- Contoh salah satu (main) data type
   data SrtSubtitle = SrtSubtitle
       { index       :: Int
       , start       :: Timestamp
       , end         :: Timestamp
       , coordinates :: Maybe SrtSubtitleCoordinates
       , taggedText  :: [TaggedText]}
     deriving (Show, Read)
   ```

   Hal menarik lain, adalah penggunaan Applicative Functor seperti `<$>` yang merupakan *infix operator* untuk fmap. Juga Applicative Typeclass `<*>` yang perlu didefinisikan bersamaan dengan deklarasi `pure` sebagai prasyarat bisa digunakannya fmap.

Referensi

1. https://hackage.haskell.org/package/base-4.15.0.0/docs/Text-ParserCombinators-ReadP.html
2. https://medium.com/@lettier/your-easy-guide-to-monads-applicatives-functors-862048d61610
3. http://learnyouahaskell.com/functors-applicative-functors-and-monoids#applicative-functors
4. https://en.wikibooks.org/wiki/Haskell/do_notation