

Języki i paradygmaty programowania (Info, III rok) 16/17

Kokpit ► Moje kursy ► JiPP.INFO.III.16/17 ► 3.4-9.4 ► Lab Skladnia 1

Lab Skladnia 1

Przygotowanie

• Przy pomocy Text.ParserCombinators.Parsec uruchomić pNat, pInt z wykładu, przy czym

```
sat = satisfy
item = anyChar
```

Uruchomić parser dla wyrażeń arytmetycznych z wykładu

Zadanie 1

Dana składnia abstrakcyjna wyrażeń arytmetycznych (jak w 2. tygodniu)

```
data Exp

= EInt Int -- stała całkowita

| EAdd Exp Exp -- e1 + e2

| ESub Exp Exp -- e1 - e2

| EMul Exp Exp -- e1 * e2

| EVar String -- zmienna

| ELet String Exp Exp -- let var = e1 in e2
```

a. zaprojektuj składnię konkretną Sugestie: standardowa notacja infiksowa oraz notacja prefiksowa a la Lisp: (* (+ 1 2) 3)

b. napisz parser dla tej składni przy uzyciu Text.ParserCombinators.Parsec

UWAGA: Ze względów wydajnościowych, operator (<|>) z biblioteki Parsec jest prawie deterministyczny i nie będzie działać dobrze dla produkcji, które mają wspólny (niepusty) prefiks.

Możemy odzyskać niedeterminizm przy pomocy kombinatora try, np.

```
try p <|> q
```

Zadanie 2

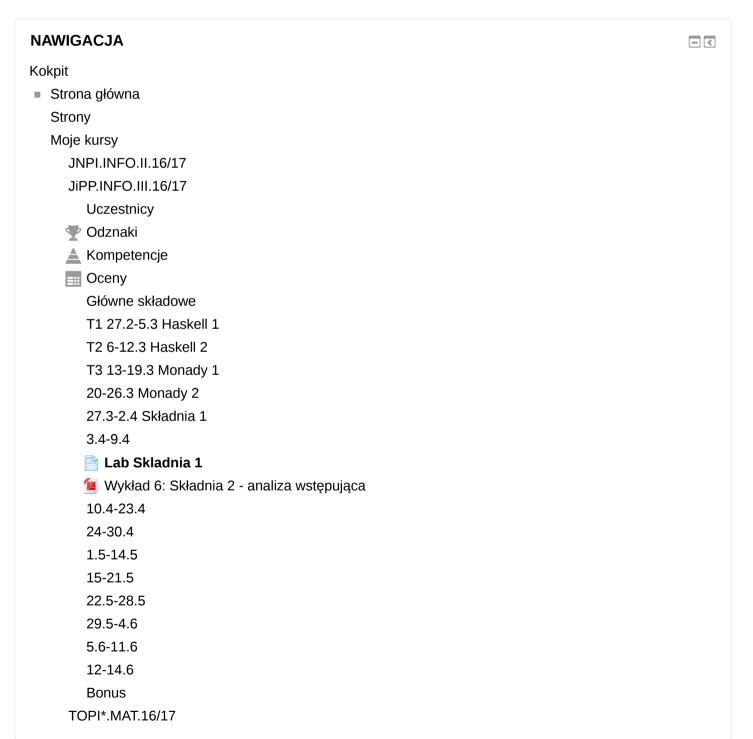
Napisz własne wersje kombinatorów parsujących użytych w poprzednim zadaniu

Sugestie:

```
newtype Parser a = Parser { runParser :: String -> [(a,String)] }
newtype Parser a = Parser {
 runParser :: String -> [Either ErrorMessage (a,String)]
}
albo, używając transformatorów monad

type Parser a = StateT [Char] (ErrorT String []) a

Ostatnia modyfikacja: czwartek, 23 kwiecień 2015, 09:36
```



ADMINISTRACJA