Декларативное программирование

Весна 2023, семинар №7

Завьялов А.А.

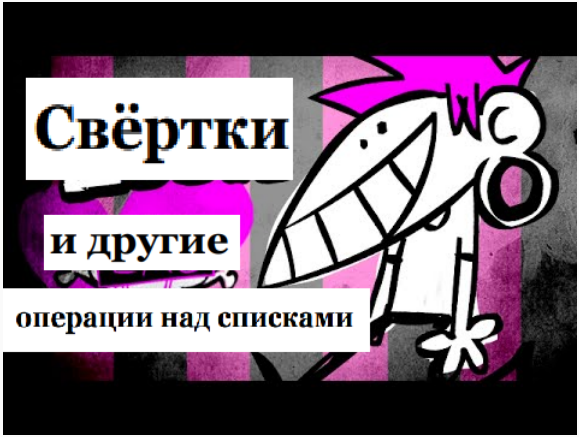
21 марта 2023 г.

Кафедра систем информатики ФИТ НГУ

А помните?...

А помните?...

А помните?...



Класс типов Foldable

```haskell
class Foldable t where
  {-# MINIMAL foldMap | foldr #-}
  foldMap :: Monoid m => (a -> m) -> t a -> m
  foldMap f = foldr (mappend . f) mempty

  foldr :: (a -> b -> b) -> b -> t a -> b
  foldr f z t = appEndo (foldMap (Endo . f) t) z
```

где

```haskell
  newtype Endo a = Endo { appEndo :: a -> a }
```

Полезные функции Data.Foldable

```
toList :: Foldable t => t a -> [a]

null :: Foldable => t a -> Bool

length :: Foldable t => t a -> Int

elem :: (Eq a, Foldable t) => a -> t a -> Bool

maximum :: (Ord a, Foldable t) => t a -> a

sum, product :: (Num a, Foldable t) => t a -> a
```

Список это Foldable

```
instance Foldable [] where
    elem    = List.elem
    foldl   = List.foldl
    foldr   = List.foldr
    length  = List.length
    maximum = List.maximum
    product = List.product
```

Реализуем Foldable

```haskell
instance Foldable (Either a) where
    foldMap _ (Left _) = mempty
    foldMap f (Right y) = f y

    foldr _ z (Left _) = z
    foldr f z (Right y) = f y z

    length (Left _)  = 0
    length (Right _) = 1

    null             = isLeft
```

Foldable и коллекции

```
instance Foldable NonEmpty
instance Foldable Set
instance Foldable (Map k)
instance Foldable (Array i)
instance Foldable Vector
```

Класс типов Traversable

Документация: “functors representing data structures that can be traversed from left to right”.

```
class (Functor t, Foldable t) => Traversable t where
  traverse :: Applicative f => (a -> f b) -> t a -> f (t b)
  traverse f = sequenceA . fmap f

  sequenceA :: Applicative f => t (f a) -> f (t a)
  sequenceA = traverse id
  {-# MINIMAL traverse | sequenceA #-}
```

Пример

```haskell
instance Travesable Maybe where
  traverse _ Nothing = Nothing
  traverse f (Just x) = Just <$> f x

instance Traversable [] where
  traverse g = foldr consF (pure [])
    where
      consF x ys = liftA2 (:) (g x) ys
```

Полезные функции Data.Traversable

```
mapM :: Monad m => (a -> m b) -> t a -> m (t b)

sequence :: Monad m => t (m a) -> m (t a)

forM :: (Traversable t, Monad m) => t a -> (a -> m b) -> m (t b)

for :: (Traversable t, Applicative f) => t a -> (a -> f b) -> f (t b)
```

Ещё больше полезных функций Data.Foldable

```
traverse_ :: (Foldable t, Applicative f) => (a -> f b) -> t a -> f ()

for_ :: (Foldable t, Applicative f) => t a -> (a -> f b) -> f ()

sequenceA_ :: (Foldable t, Applicative f) => t (f a) -> f ()

mapM_ :: (Foldable t, Monad m) => (a -> m b) -> t a -> m ()

forM_ :: (Foldable t, Monad m) => t a -> (a -> m b) -> m ()

sequence_ :: (Foldable t, Monad m) => t (m a) -> m ()
```

Класс типов Alternative

Аппликативный функтор, являющийся *моноидом*:

```
class Applicative f => Alternatve f where
  empty :: f a
  (<|>) :: f a -> f a -> f a
{-# MINIMAL empty, (<|>) #-}

infix 3 <|>
```

Класс типов MonadPlus

Почти как `Alternative`, но для монад:

```
class (Alternative m, Monad m) => MonadPlus m where
  mzero :: m a
  mplus :: m a -> m a -> m a
```

Законы MonadPlus

- `mzero >>= f == mzero`
- `v >> mzero == mzero`
- `https://wiki.haskell.org/MonadPlus`

MonadPlus – пример

```haskell
mfilter :: (MonadPlus m) => (a -> Bool) -> m a -> m a
mfilter p ma = do
  a <- ma
  if p a then return a else mzero

guard :: Alternative f => Bool -> f ()
guard True = pure ()
guard False =  empty

when :: Applicative f => Bool -> f () -> f ()
when p s  = if p then s else pure ()
```

Монада Cont

Представляет вычисления в стиле передачи продолжений (Continuation Passing Style, CPS[1]). В таком стиле результат функции не возвращается, а передается в другую функцию как *параметр*.

```
class Monad m => MonadCont m where
  callCC :: ((a -> m b) -> m a) -> m a

newtype Cont r a = Cont { runCont :: (a -> r) -> r }

instance Monad (Cont r) where
    return a = Cont ($ a)
    m >>= k  = Cont $ \c -> runCont m $ \a -> runCont (k a) c
```

[1] https://en.wikipedia.org/wiki/Continuation-passing_style

Q&A