# Трансформеры монад

## Монады

```
newtype Reader r a =
    Reader { runReader :: r -> a }
newtype Writer w a =
    Writer { runWriter :: (a, w) }
newtype State s a =
    State { runState :: s -> (a, s) }
```

# Трансформеры монад

```
newtype ReaderT r m a =
    ReaderT { runReaderT :: r -> m a }
newtype WriterT w m a =
    WriterT { runWriterT :: m (a, w) }
newtype StateT s m a =
    StateT { runStateT :: s -> m (a, s) }
```

# Трансформеры монад

```
newtype ExceptT e m a =
    ExceptT { runExceptT :: m (Either e a) }
newtype MaybeT m a =
    MaybeT { runMaybeT :: m (Maybe a) }
```

## Порядок

```
embedded :: MaybeT
              (ExceptT String
                 (ReaderT () IO)) Int
embedded = return 1

maybeUnwrap :: ExceptT String (ReaderT () IO) (Maybe Int)
maybeUnwrap = runMaybeT embedded

eitherUnwrap ::ReaderT () IO (Either String (Maybe Int))
eitherUnwrap = runExceptT maybeUnwrap

readerUnwrap :: () -> IO (Either String (Maybe Int))
readerUnwrap = runReaderT eitherUnwrap

ghci> readerUnwrap ()
Right (Just 1)
```

## lift

```
import Control.Monad.Trans.Class


class MonadTrans t where
	lift :: (Monad m) => m a -> t m a
```