А. Б. Волотовский
(*ГГУ им. Ф. Скорины, Гомель*)

ТРАНСФОРМЕРЫ МОНАД НА HASKELL

В функциональном программировании программа представляет собой композицию функций. Однако функции с вычислительными *эф-*

фектами не имеют тривиальной логики композиции, что решается *монадами* – абстракциями над цепочками таких вычислений. На рисунке 1 показано, как разнородные вычисления сводятся к общему виду.

# Псевдокод №1.	# Псевдокод №2.	# Монадическое
$x_1 = action_1$	$xs_1 = action_1$	#обобщение,
$\boldsymbol{if}\ success$	$\boldsymbol{for}\ x_1\ \boldsymbol{in}\ xs_1$	# упростившее
$\quad x_2 = action_2$	$\quad xs_2 = action_2$	#псевдокод.
$\quad \boldsymbol{if}\ success$	$\quad \boldsymbol{for}\ x_2\ \boldsymbol{in}\ xs_2$	$x_1 \leftarrow action_1$
$\quad\quad \boldsymbol{return}\ (x_1 + x_2)$	$\quad\quad \boldsymbol{yield\ return}\ (x_1 + x_2)$	$x_2 \leftarrow action_2$
$\boldsymbol{throw}\ error$		$\boldsymbol{return}\ (x_1 + x_2)$

Рисунок 1 – Монадическое обобщение вычислений

Обычно в приложениях вычисления имеют несколько эффектов сразу. Для этого используют *трансформеры* [1, с. 36], объединяющие одни монады с другими. Тогда, например, монада для взаимодействия с API ВКонтакте будет иметь вид

$$ApiM = ReaderT\ ApiConnection\ (ExceptT\ ApiError\ IO).$$

Каждый трансформер определяет функцию $lift$, поднимающую вычисления из внутренней монады на уровень его самого. В случае нескольких монад, код засоряется частыми $lift$. Для повышения читаемости используем шаблон [1, с. 28], состоящий из 3-х этапов:

1. Описание интерфейса разрабатываемой монады.

В случае API ВКонтакте это будет:

$$\begin{aligned} &\boldsymbol{class}\ MonadIO\ m\ \Rightarrow MonadApi\ m\ \boldsymbol{where} \\ &\quad askApiConnection\ :: m\ ApiConnection \\ &\quad throwApiError\ \ :: ApiError \rightarrow m\ a \end{aligned}$$

2. Описание конкретной монады. См. (1).

3. Реализация интерфейса для конкретной монады, куда будут помещены все использования функции $lift$.

Описанный способ применения монад и их трансформеров, поможет разработчику скрыть часть логики в реализациях сложных алгоритмов (монотонную обработку ошибок, обход каждого результата при недетерминированных вычислениях и т.п.).