Зертханалық жұмыс №5

Haskell тілінде тізімдер мен жолдарды өңдеу

λ есептеуді қолдану

Программалаудың функционалдық парадигмасы λ-есептеуге негізделгендіктен, барлық функционалдық тілдердің λ-абстракцияны сипаттауға арналған қағиданы қолдайтыны заңды құбылыс. Haskell бұл аспектіні де айналып өткен жоқ, қандай да бір функцияны анықтауды, қажет болса λ-абстракция арқылы жүргізеді. Сонымен қатар λ-абстракция арқылы жасырын функцияларды да анықтауға болады (Мысалы, бірлік шақыру үшін). Төменде, λ-есептеулер көмегімен add және inc функцияларын анықтаудың мысалы келтірілген.

Мысал 2. λ абстракциялар арқылы анықталған add және inc функциялары.

```
add = \x y > x + y
inc = \x > x + 1
```

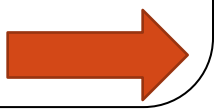
Мысал 3. Жасырын функцияны шақыру.

```
cubes = map (\x > x * x * x) [0 ..]
```

Мысал 3 берліген параметрдің кубын есептейтінжасырын функцияны шақыруды көрсетіп тұр. Бұл инструкцияның орындалу нәтижесі, нөлден басталған бүтін сандар кубтарының шексіз тізімі болады. Ескертетін жағдай, Haskell'де өрнектерді жазудың қысқартылған тәсілі қолданылады, себебі тура қағида бойынша add функциясын келесі түрде жазған дұрысырық болар еді:

```
add = \x > \y > x + y
```

λ абстракцияның типін анықтау, функцияның типін анықтағандай жүргізіледі. λ x.expr түріндегі λ өрнектің типі келесі түрде болады T1 → T2, мұндағы T1 - x айнымалысының типі, ал T2 —expr өрнегінің типі.



Функцияны жазудың инфиксті тәсілі

Кейбір функциялар үшін жазудың инфиксті тәсілі қолданылуы мүмкін, ондай функциялар қарапайым бинарлық операциялар сияқты. Мысалы, тізімдерді конкатенациялау мен функцияларды композициялау операциялары анықталған:

Мысал 4. Тізімдерді конкатенациялаудың инфиксті операциясы.

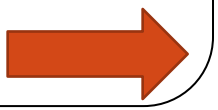
```
(++)                :: [a] > [a] > [a]
[] ++ ys            = ys
(x:xs) ++ ys        = x : (xs ++ ys)
```

Мысал 5. Функцияны композициялаудың инфиксті операциясы.

```
(.)     :: (b > c) > (a > b) > (a > c)
f . g   = \x > f (g x)
```

Haskell'-де инфиксті операциялар функция болып табылатындықтан, олар каррирленген, сондықтан мұндай функцияларды бөліктеп қолдану мүмкіндігін қамтамасыз еткен маңызды. Осы мақсатта, Haskell'де «секция» деп аталатын арнайы жазба қолданылады:

```
(x ++)      = \x > (x ++ y)
(++ y)      = \y > (x ++ y)
(++)        = \x y > (x ++ y)
```



Жоғарыда үш секция көрсетілген, олардың әрқайсысы берілген аргументтер санына сәйкес тізімдерді конкатенциялаудың инфиксті операцияларын анықтайды. Секцияны жазуда дөңгелек жақшаларды қолдану міндетті болып табылады. Егер қандай да бір функция екі параметр қабылдайтын болса, онда оны да инфиксті түрде жазуға болады. Бірақ, жай ғана параметрлер арасына функция атауын жазу – қате деп есептеледі, себебі Haskell' қағидасы бойынша екі рет қолдану болып қалады және бір қолдануда бір операнд жетіспей қалады. Функцияны инфиксті түрде жазу үшін, оны кері апостроф — ` символына алу қажет.

Қайта анықталған инфиксті операциялар үшін есептеу ретін анықтау мүмкін. Ол үшін Haskell'де infixr қызметші сөзі қолданылады. Қызметші сөз берілген операцияның 0 ден 9-ға дейінгі аралықтағы маңыздылық дәрежесін (орындалу ретін) айқындайды және 9 мұндағы ең жоғарғы маңыздылық дәрежесі болып табылады. 4 және 5 мысалдардағы берілген операциялардың дәрежесі келесі түрде анықталады:

infixr 5 ++
infixr 9 .

Ескертетін жай, Haskell'дегі барлық функциялар қатаң емес, яғни олар есептеулерді кейінге қалдырып есептей алады. Мысалы, егер қандай да бір функция балай анықталса:

```
bot = bot
```

Мұндай функцияны шақыру кезінде қате кетеді және ондай қателерді табу қиынға соғады. Бірақ қандай да бір тұрақты функция болса және келесі түрде анықталса:

```
constant_1 x  = 1
```

Онда (constant_1 bot) конструкциясын шақыру кезінде ешқандай қате кетпейді, себебі бұл жағдайда bot функциясының мәні есептелмейді (есептеу кейінге қалдырылады, өрнектің мәні тек қажет болған жадайда ғана есептеледі). Есептеу нәтижесі 1 саны болатыны анық көрініп тұр.