

Зертханалық жұмыс N°2

Haskell тілінің қарапайым функциялары



Деректер құрылымы мен олардың типтері

Кез-келген программалау тілінің базалық бірлігі – символ. Символ дегеніміз ұзындығы шектеулі немесе шектеусіз әріптер, символдар және арнайы белгілер тізбегі. Кейбір тілдерде кіші және бас әріптердің айырмашылығы болса, кейбірінде болмайды. Мысалы, **Lisp**'те кіші және бас әріптердің айырмашылығы жоқ болса, **Haskell**'де бар.

Символдар көбінесе идентификаторлар – тұрақты, айнымалы, функция аттары ретінде қолданылады. Тұрақты, айнымалы және функциялар таңбалардың типтелген тізбегі болып табылады. Әріптер қатары сандық константаның мәні бола алмайды. Функционалдық тілдерде атом – базалық түсінігі кездеседі. Іс жүзінде атомдар дегеніміз символдар мен сандар.

Функционалдық программалаудың келесі түсінігі – тізімдер. Абстрактілі математикалық қағидада $[]$ символдары қолданылды, ол **Haskell**'де де қолданылады. Бірақ **Lisp**'те кәдімгі «дөңгелек» жақшалар қолданылады — $()$. **Lisp**'те тізім элементтері бос орын арқылы ажыратылса **Haskell**'де элементтерді ажырату үшін үтір қолданылады. Сонда, $[a, b, c]$ тізімі **Haskell**' синтаксисі бойынша осылай жазылса, **Lisp**' қағидасы бойынша $(a\ b\ c)$ түріне аудару керек. Бірақ **Lisp**' ті жасаушылар жұптарды ұйымдастыру үшін нүктелік жазбаны да қолданып, жоғарыдағы тізімді келесі түрде жазды $(a.(b.(c.NIL)))$.

Lisp' пен **Haskell**'де тізімдік құрылымдар - бір тізімді екіншісіне алу қағидасы бойынша сипатталады. **Lisp**' қағидасы бойынша, ішкі тізім жақшасының алдына бос орын қалдырмаса да болады.

Функционалдық тілдерде деректер типі автооматты түрде анықталады. Типті автоматты түрде анықтау механизмі **Haskell** тіліне де енгізілген. Бірақ, кей жағдайда типті көрсету қажет болады, әйтпесе интерпретатор шатасып кетуі мүмкін. **Haskell**'де арнайы символ қолданылады — $::$ (екі қос нүкте), ол «типі бар» деп оқылады.



Егер

5 :: Integer

Бірақ Haskell полиморфты типтер, немесе типтер шаблонын да қолдайды. Мысалы, [a] түрінде жазсақ, онда тип «кез-келген тип атомдарының тізімі» деп оқылады, және атомдар типі барлық тізім құру барысында бірдей болу керек. Яғни, [1, 2, 3] және ['a', 'b', 'c'] тізімдері [a] типті, ал [1, 'a'] тізімінің типі басқа болады.

Атау беру бойынша келісімдер

Haskell'де атау беру бойынша келісімдер өте маңызды, себебі олар тілдің синтаксисіне енеді. Ең басты келісім – идентификатор бас әріптен басталуы керек. Тип атаулары да, бас әріптерден басталуы керек. Функциялардың, айнымалы мен тұрақтылардың атаулары кіші әріптерден басталады. Идентификатордың бірінші әріпі ретінде арнайы таңбалар да алынуы мүмкін.

Тізім анықтауыштары мен математикалық тізбектер.

Haskell — қарапайым математикалық формула арқылы оңай, әрі жылдам тізімдер құруға мүмкіндік беретін жалғыз программалау тілі. Бұл тәсіл тізімді жылдам сұрыптау функциясын Хоар әдісімен құру кезінде қолданылған болатын. Тізімді анықтауыштардың жалпы жазылуы келесі түрде болады:

[x | x < xs]

Бұл жазу былай оқылады «xs-тен алынған барлық x-тер тізімі». «x ← xs» құрылымы генератор деп аталады. Мұндай генератордан кейін үтірмен ажыратылған қандай да бір өрнек мәні тұру керек. Сонда барлық осындай x-тер таңдалып, барлық өрнек мәні үшін орындалады. Жазу келесі түрде болады:

[x | x < xs, x > m, x < n]

Можно прочитать как «Список из всех таких x, взятых из xs, что (x больше m) И (x меньше n)».

Haskell'дің тағы бір артықшылығы шексіз тізімдер мен деректер құрылымын құрудың оңайлығы болып табылады. Шексіз тізімдерді тізім анықтауышы негізінде де, арнайы қағида көмегімен де құруға болады. Мысалы, төменде натурал сандар тізбегінен тұратын шексіз тізім көрсетілген. Екінші тізім тақ натурал сандардың тізбегін құрайды.

[1, 2 ..]

[1, 3 ..]



Екі нүкте көмегімен кез-келген шексіз және шектеулі арифметикалық прогрессияларды да анықтауға болады. Егер тізбек шектеулі болса, онда оның бірінші және соңғы элементтері беріледі. Арифметикалық прогрессияның айырмасы бірінші және екінші берілген элементтер негізінде есептеледі – жоғарыда келтірілген мысалдарда бірінші прогрессиядағы айырма – 1, ал екіншісінде – 2 болады. Сонымен, 10-ға дейінгі барлық тақ натурал сандардың тізімін анықтау үшін келесі тізімді жазу керек: [1, 3 .. 10]. Нәтижесі төмендегі тізім болады: [1, 3, 5, 7, 9].

Деректердің шексіз құрылымын шексіз тізімдер негізінде де, рекурсия механизмін қолдану арқылы да анықтауға болады. Бұл жағдайда рекурсия рекурсивті функцияларға қатынау түрінде қолданылады. Шексіз деректер құрылымын құрудың үшінші тәсілі шексіз типтерді қолдану арқылы жүргізіледі.

Мысал 1. Екілік бұтақты бейнелейтін типті анықтау.

data Tree a

Branch

Leaf

= Leaf a

| Branch (Tree a) (Tree a)

: Tree a > Tree a > Tree a

> Tree a



Бұл мысалда шексіз типті анықтау тәсілі көрсетілген. Рекурсия қолданылғаны көрініп тұр. Егер деректердің жаңа типін құру қажет болмаса, шексіз құрылымды функцияның көмегімен де алуға болады:

```
ones
```

```
numbersFrom n
```

```
= 1 : ones
```

```
= n : numberFrom (n + 1)
```

```
squares
```

```
= map (^2) (numbersFrom 0)
```

Бірінші функция бірліктерден тұратын шексіз тізбекті анықтайды. Екінші функция берілген саннан бастап, барлық бүтін сандар тізбегін құрады. Үшінші функция нөлмен бірге натурал сандар квадраттарының шексіз тізбегін құрады.