Вивчить собі Хаскела на велике щастя!

Автор: Міран Ліповача

Переклад здійснили: Ганна Лелів, Семен Тригубенко, Богдан Пеньковський, Марина Стрельчук і Тетяна Богдан

Мовні редактори: Тетяна Богдан і Ганна Лелів
Науковий редактор: Семен Тригубенко

Переклад виконано за підтримки
Словненя

2017-05-21T00:06:09Z
Версія v4.7-54-gda41cf2

Зміст

Розділ 1

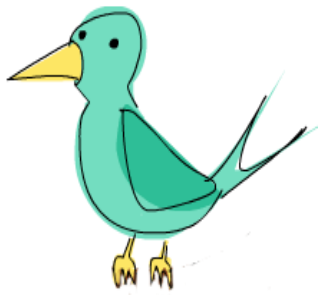
Передмова

Переклад українською Тетяни Богдан

1.1 Про цей підручник

Ласкаво просимо до **Вивчить собі Хаскела на велике щастя!** Якщо ви читаєте ці рядки тексту, то напевно хочете вивчити Хаскел... І, справді, ви прийшли туди, куди треба! Але давайте відкладемо навчання на хвильку, і спершу поговорімо про зміст цього посібника.

Я вирішив написати цей підручник аби закріпити власне володіння Хаскелом, а також аби допомогти тим, хто не знайомий з Хаскелом, подивитися на нього крізь призму мого досвіду. В Інтернеті тиняється досить багато посібників з Хаскела. Коли я починав вчитися, я користувався кількома джерелами. На кожну тему я читав декілька посібників чи статей, які пояснювали один і той же матеріал по-різному. Скориставшись кількома джерелами, я міг скласти окремі деталі в одне ціле. І раптом усе ставало зрозумілим. Отож, цей посібник — це спроба створити ще один корисний ресурс з вивчення Хаскела, щоб читачі мали змогу підібрати посібник на свій смак.



Цей посібник для тих, хто має досвід програмування імперативними мовами (C, C++, Java, Python...), але ще не стикався з функційними мовами програмування (Haskell, ML, OCaml...). Хоча я певен, що навіть без серйозного досвіду програмування така розумна людина, як мій читач, зможе розібратися і вивчити Хаскел!

Канал #haskell в мережі freenode — це чудовий ресурс для тих, хто застряг на якомусь питанні і хоче

про щось запитати. Люди там дуже добрі, терплячі і ставляться до новачків з розумінням.

Поки я нарешті збагнув, що таке Хаскел, я двічі зазнав поразки, намагаючись його вивчити, тому що все здавалось страшенно дивним, і я ніяк не міг зрозуміти, що до чого. Але одного дня все раптом «розвиднілось», а коли я подолав перші перешкоди, усе пішло, як по маслу. Я веду до того, що Хаскел — це класна мова, і якщо вас справді цікавить програмування, то цю мову варто вивчити, навіть якщо все спочатку здається дивним. Учити Хаскел — це як уперше вчитися програмувати — це прикольно! Хаскел примушує вас думати по-новому, і саме про це йдеться у наступному розділі...

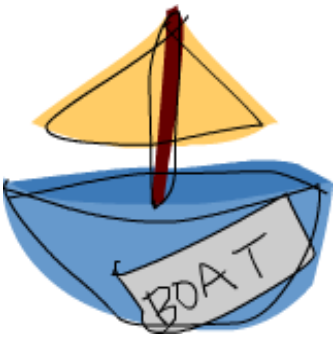
1.2 Отже, що воно таке Хаскел?

Хаскел — це **чистофункційна мова програмування** [purely functional programming language]. В імперативних мовах ви даєте комп'ютеру перелік завдань, і він їх виконує. Виконуючи завдання, він може змінювати стан. Наприклад, ви задаєте змінній `a` значення 5, виконуєте якусь операцію, і задаєте їй інше значення. Є також засоби керування потоком, — наприклад, для виконання певної операції декілька разів. У чистофункційному програмуванні комп'ютеру не кажуть, що робити, а описують, що ϵ що. Факторіал числа — це добуток усіх чисел від одиниці до того числа, сума списку чисел — це перше число плюс сума решти чисел, і так далі. Операції виражаються в формі функцій. Також не можна задавати змінній одне значення, а потім змінювати його на інше. Якщо ви сказали, що `a` дорівнює 5, то потім не можете сказати, що вона дорівнює чомусь іншому, бо ви щойно сказали, що це 5. Ви ж не брехун/брехуха, чи не так? Отже, у чистофункційних мовах програмування функція не має побічних ефектів. Все, що може функція, — це щось порахувати і повернути результат. На перший погляд здається, що функція має обмежені можливості, але насправді ефект просто чудовий: якщо двічі викликати функцію з одними й тими ж параметрами, вона гарантовано поверне один і той же результат. Ця поведінка називається **прозорістю посилань** [referential transparency], і вона не тільки дозволяє компіляторові розмірковувати про поведінку програми, а й легко відстежити (чи навіть довести), що функція правильна, а тоді будувати складніші функції, склеюючи прості функції докупи.

Хаскел — **лінивий** [lazy]. Тобто якщо йому нічого не наказати, Хаскел не виконуватиме функції та не робитиме обчислень до тих

пір, поки його не змусять показати результат. Це чудово поєднується з прозорістю посилань і дозволяє думати про програми як про низки **перетворень даних** [transformations on data]. Це також уможливлює існування нескінченних структур даних. Скажімо, у вас є **незміненний** [immutable] список чисел `xs = [1,2,3,4,5,6,7,8]` і функція `doubleMe`, яка множить кожен елемент на 2 і повертає новий список. Якщо ви помножите список на 8, імперативною мовою, виконавши `doubleMe(doubleMe(doubleMe(xs)))`, то програма пройдеться списком один раз, зробить копію і поверне її як результат. Тоді вона пройдеться тим списком ще пару разів, і поверне результат. Натомість у лінивій мові на ваше прохання виконати `doubleMe` без виводу результату програма фактично відповідає: «Ага, при нагоді порахую!». Але щойно ви захочете побачити результат, перша `doubleMe` скаже другій, що вимагає результату негайно! Друга це передає третій, і третя неохоче повертає подвоєну 1, тобто 2. Друга функція отримує її і повертає 4 першій. Перша, своєю чергою, сповіщає вам, що перший елемент списку дорівнює 8. Отже, програма проходиться списком лише один раз і лише тоді, коли це справді потрібно. Таким чином, коли ви хочете щось зробити в лінивій мові, просто візьміть початкові дані і успішно перетворюйте їх доти, поки не отримаєте потрібний результат.

Хаскел — **статично типізований** [statically typed]. Обробляючи програму, компілятор знає, який шматок коду є числом, який — рядком, і так далі. Тобто під час компіляції ви знайдете чимало можливих помилок. Якщо ви спробуєте додати число і рядок, компілятор вас насварить. Хаскел використовує дуже гарну систему типів, яка підтримує **виведення типів** [type inference]. Це означає, що не обов'язково вказувати тип кожного шматку коду, система типів достатньо розумна, щоб самій про багато що здогадатися. Якщо ви написали `a = 5 + 4`, то не треба зазначати, що `a` — це число, Хаскел сам це зрозуміє. Також, завдяки виведенню типів ваш код стає більш загальним. Якщо функція приймає два параметри і додає їх один до одного, то не треба явно вказувати їхній тип, адже функція здатна приймати два будь-яких параметри, які поводяться як числа.



Хаскел **елегантний і стислий**. Оскільки Хаскел — це високорівнева мова програмування, написані на ньому програми коротші за їхні імперативні еквіваленти. А коротші програми легше підтримувати, і в них трапляється менше

помилок.

Хаскел був розроблений **дуже розумними хлопцями** (з кандидатськими ступенями). Розробка Хаскела почалась в 1987 році, коли комітет розробників зібрався аби створити таку собі некепську мову. В 2003 було опубліковано Haskell Report, який означив стабільну версію мови.

1.3 Що треба, щоб негайно узятися до роботи

Текстовий редактор і компілятор Хаскела. У вас вже мабуть є улюблений текстовий редактор, тому не будемо на цьому зупинятися. У цьому посібнику ми використовуватимемо GHC, найпоширеніший компілятор Хаскела. Найпростіше завантажити Haskell Platform — компілятор з усім необхідним.

GHC читає файли із сирцями Хаскела (вони здебільшого мають розширення .hs) і компілює їх, а також має інтерактивний режим, в якому можна інтерактивно взаємодіяти з програмою. Інтерактивно! Можна викликати функції з підвантажених програм і одразу ж бачити результат обрахунку. Для навчання це набагато швидше і зручніше, аніж компілювати і запускати програму після кожної зміни. Інтерактивний режим викликається командою **ghci**. Якщо означити якісь функції у файлі **myfunctions.hs**, ці функції можна завантажити командою **:l myfunctions** і тоді бавитися з ними, за умови, що файл **myfunctions.hs** знаходиться в тій самій директорії, з якої був запущений **ghci**. Якщо .hs змінився, його можна перезавантажити командою **:l myfunctions** або **:r**, яка перевантажує поточну програму. Як правило, я так і працюю: означую певні функції в якомусь .hs файлі, завантажую файл, бавлюся з функціями, а потім змінюю .hs файл і так далі. Надалі ми робитимемо так само.

Покажчик