## Языки для математиков (продолжение)

Язык Haskell имеет два типа файлов с расширениями hs или lhs. В файлах второго типа комментарии и операторы программы меняются ролями. Содержимое файла – произвольный текст, а строки с определениями функций начинаются со знака «>» и отделяются от основного текста пустыми строками.

Программы на Haskell в 4-5 раз короче аналогичных, написанных на традиционных языках. Кроме того, как правило, эти программы к тому же будут ограничены по типам и размерам данных. В Haskell списки могут быть даже бесконечными!

Ниже приведено содержимое файла с расширением lhs. Первая половина строк – это условия задач, вторая (18 строк) – их решение. Когда спрашиваешь студентов, сколько строк будет содержать программа на каком-либо языке (например, Паскаль или C#), в ответ говорят, что строк 150-200. Как оказалось, оценка правильна.

Программа для решения этих задач на Компонентном Паскале заняла у меня 218 строк, из которых 47 – это тестирующий модуль. При этом длина многочленов ограничена некоторой константой, а значения коэффициентов должны находиться в диапазоне от $-2^{-31}$ до $2^{31}-1$. Её можно было бы «ужать» в ущерб читабельности, но все равно размер окажется раз в 5 больше.

```
Многочлен n-й степени с целыми коэффициентами представлен списком
своих коэффициентов, начиная со старшего члена.
Например,  многочлен x^3+3x-2  представляется списком [1,0,3,-2]
Разработать консольные программы для следующих задач:
1. Найти сумму двух многочленов одной степени.
2. Найти сумму двух многочленов различных степеней.
3. Умножить многочлен на заданное число.
4. Вычислить значение многочлена для заданного x по схеме Горнера.
5. Найти значение коэффициента при x^n в произведении многочлена n-й
   степени на другой многочлен (возможно, другой степени).
6. Вычислить скалярное произведение двух векторов.
7. Вычислить произведение двух многочленов.
8. Построить многочлен с целыми коэффициентами с заданными корнями.
9. Найти все делители (положительные и отрицательные) целого числа.
10. Найти все целочисленные корни многочлена с целыми коэффициентами.
11. Найти производную многочлена.
12. Получить значения многочлена на отрезке [0..n] в целых точках.

> pAdd0 m1 m2 | length m1 == length m2 = zipWith (+) m1 m2
>             | otherwise = error "different length"
> pAdd m1 m2  | k >=0     = pAdd0 m1 ((replicate k 0)++m2)
>             | k < 0     = pAdd0 ((replicate (-k) 0)++m1) m2
>             where k     = length m1 - length m2
> polyN m n = map (*n) m
> polyX m x = foldl f 0 m where f a b = a*x+b
> kn m1 m2  = sum (zipWith (*) m1 (reverse m2))
> sv x y    = sum (zipWith (*) x y)
> mms x []  = []
> mms x y   = (sv x y) : (mms x (tail y))
> pMult x y = mms (reverse x) ((replicate (length x - 1) 0) ++ y)
> pSetr kk  = foldl pMult [1] [[1,-k] | k<-kk]
> delp n    = [k | k<-[1..n], mod n k == 0]
> delims n  = d1 ++  map (\x -> -x) d1 where d1 = delp n
> pRoots m  = [k | k<- delims (last m), polyX m k == 0]
> pDer m = reverse (tail (zipWith (*) [0..length m -1] (reverse m)))
> pVal m n  = map (polyX m) [0..n]
```

В решении используется 10 функций из стандартного модуля Prelude, который импортируется по умолчанию. Некоторые функции очень простые:

`Length` – длина списка,

`otherwise` – функция со значением false,

`error` – прерывание программы и вывод сообщения,
`replicate` – дублирование списка заданное число раз,
`sum` – сумма элементов списка,
`reverse` – «переворачивание» списка,
`tail` – список без первого элемента,
`map` – применение функции к списку.

Осталось пояснить, что делают функции `zipWith` и `foldl`.

Есть функция `zip`, которая склеивает два списка, образую список пар, например,

```
> zip [1,2] [3,4]
[(1,3),(2,4)]
```

Функция `zipWith` сначала делает то же самое, а затем к каждой паре применяет заданную функцию. Например,

```
> zip (*) [1,2] [3,4]
[3,8]
```

Функция `foldl` «сворачивает» список в одно значение, используя заданную функцию. Например, вызов

```
> foldl (*) 2 [3,4,5]
```

работает так. Сначала 2 умножается на первый элемент списка и аргументами становятся новое значение второго параметра и укороченный список

```
foldl (*) 6 [4,5]
```

И так повторяется до тех пор, пока список не опустеет:

```
foldl (*) 24 [5]
120
```

С помощью этой функции просто определить функцию факториала:

```
fact n = foldl (*) 1 [1..n]
```

В стандартном модуле Prelude около 300 функций, которых достаточно для решения большого класса задач. Кроме того в стандарт входит еще около 100 модулей, сгруппированных в 6 пакетов [2]. Трудно найти область применения, для которой нельзя было бы подыскать подходящие наборы модулей.

Когда я познакомился с языком Haskell, а произошло это с появлением первой книги на русском языке [1], меня сразу же это увлекло. С тех пор я использовал язык для написания различных небольших приложений, каждый раз поражаясь выразительностью и мощностью языка.

Давно зная о проекте Эйлер (https://projecteuler.net/ и зеркало на русском http://euler.jakumo.org/), на котором 761 задача для программирования, я попробовал свои силы в их решении. Первые несколько задач мне любезно предоставил Р. Душкин, естественно на Haskell'е. Все последующие я также решал на Haskell'е. Решив задачу, вы попадаете в своеобразный клуб тех, кто решил эту же задачу, и можете посмотреть решения на других языках.

Первая задача (самая легкая) и ее решение.

```
-- Найдите сумму всех чисел меньше 1000, кратных 3 или 5.
problem_001 = sum [x | x <- [1..999], suitable x]
  where
     suitable n = (n `mod` 3) == 0 || (n `mod` 5) == 0
```

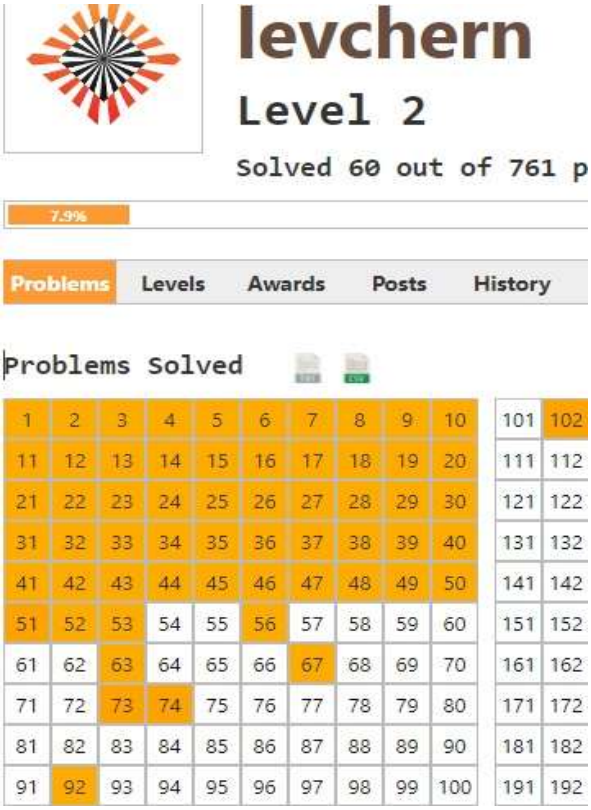
Сейчас мое достижение – 60 решенных задач (большинство в окт. 2013, последние 9 в мае 2018).

В проекте зарегистрировано 1036105 участников. Интересно, на каких языках решались эти задачи. Всего их – 106! Пять самых популярных: Python(50866), C/C++(45743), Java(29658), C#(14046), Haskell (7083). В скобках указано число участников (данные на 5.8.2021).

В проекте постоянно появляются новые задачи. Задачи очень разнообразны и требуют, как правило, нетривиальных подходов. Вот одна из последних задач:

Назовем натуральное число N *скрытным*, если существуют такие натуральные числа a, b, c и d, что ab = cd = N и a+b = c+d+1. Сколько всего существует скрытных чисел, не превышающих $10^{14}$?

Простой перебор не пройдет!




| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 101 | 102 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 111 | 112 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 121 | 122 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 131 | 132 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 141 | 142 |
| 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 151 | 152 |
| 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 161 | 162 |
| 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 171 | 172 |
| 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 181 | 182 |
| 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 191 | 192 |

Программирование на Haskell, как собственно, и на других «математических» языках, дается не всегда просто, так как требует другого мышления, чем при программировании на традиционных фон-неймановских языках. Но оно формирует очень полезное качество: умение декомпозировать задачу на более мелкие подзадачи.

Вы познакомились с очень небольшой частью возможностей языка. Есть в нем и такие вещи, которые не по зубам даже опытным программистом. Высший пилотаж – владение аппаратом так называемых *монад*. Для математиков это понятие связано с теорий категорий, которая является своего рода абстракцией над абстракциями. В стандартном модуле реализовано несколько монад, которые определяют те же списки, позволяют работать с файлами, производить консольный ввод-вывод, и другое.


1. Душкин Р.В. Функциональное программирование на языке Haskell. – М.: ДМК Пресс, 2007.
2. Душкин Р.В. Справочник по языку Haskell. – М.: ДМК Пресс, 2008. – 554 с.,ил.
3. Душкин Р. В. 14 занимательных эссе о языке Haskell и функциональном программировании. – М.: ДМК Пресс, 2011.
4. Душкин Р.В., Чернышов Л.Н. Функциональное программирование: теория и практика. – Новые информационные тех-нологии». Тезисы докладов XYI-й Международной сту-денческой школы-семинара. – М.: МГИЭМ, 2008.