Создание сложных функций

Красным отмечены последние исправления от 20-го октября.

Цель задания — создание арифметических функций с использованием широкого спектра возможностей Haskell: рекурсии и ко-рекурсии, ветвлений, охраны, замыканий и т. п.

Все задания определяются для натуральных чисел (ℕ), если не оговорено иное. Поэтому, как правило, всегда* нужно делать проверки для аргументов, и использовать функцию **error**:

```
sg' x | x == 0    = 1
      | x >  0    = 0
      | otherwise = error "Arg must be positive!"
```

*В некоторых случаях эти проверки будут происходить автоматически за счет более ранних определений.

Плюс, минус, умножить

Будем считать, что заданы две функции `inc` и `dec`. Например, следующим образом:

```
inc x | x >= 0      = x + 1
      | otherwise   = error "Arg must be positive!"
dec x | x >  0      = x - 1
      | x == 0      = 0
      | otherwise   = error "Arg must be positive!"
```

Задача 1. *Пользуясь только функциями* `inc`*,* `dec` *и рекурсией, создать функции сложения* `pls`*, вычитания* `mns` *и умножения* `mlt`*. Для вычитания определим требование:*

$$x \dot{-} y = \begin{cases} x - y, & x \geqslant y; \\ 0, & otherwise. \end{cases}$$

т. е. введенный нами минус `mns` *должен соответствовать арифметической операции «$\dot{-}$».*

Минимум и максимум двух чисел

Задача 2. *Haskell имеет встроенные функции для нахождения* **max**, **min**. *Требуется создать, используя ветвление, свои аналоги этих функций* **max**, **min** *для чисел. Возможны варианты решения с рекурсией, с использованием ранее введенных функций* `pls`*,* `mns` *и* `mlt`*.*

Частное, остаток, делимость

В данном разделе необходимо создать функции нахождения частного и остатка двух натуральных чисел при делении с остатком. Haskell содержит для этого необходимые операции (даже несколько аналогичных операций). Но необходимо создать свои аналоги, используя рекурсию и ко-рекурсию, сложение, вычитание и умножение.

Напомним, что в теории чисел для множества $\mathbb{N}$ доказана теорема:

Теорема 1 (о делении с остатком).

$$\forall m\, \forall k{\neq}0\, \exists! n\, \exists! r \quad (m = k \cdot n + r \;\&\; r < k). \tag{1}$$

В таком случае говорим, что число m делится на число k с остатком r. Число n будет называться частным.

Задача 3. *Используя ко-рекурсию, умножение, сложение и вычитание, написать функцию вычисления частного двух чисел из* `Integer`. *Для отрицательных аргументов не забывать использовать* `error`. *Деление на ноль можно доопределить как x или как 0.*

Задача 4. *Используя рекурсию, умножение, сложение и вычитание, написать функцию вычисления частного двух чисел из* `Integer`. *Для отрицательных аргументов не забывать использовать* `error`. *Деление на ноль можно доопределить как x или как 0.*

Задача 5. *Написать аналогичные функции для вычисления остатка.*

Задача 6. *Написать функцию-предикат $(x \,\vdots\, y)$ для натуральных чисел $\mathbb{N}$, исходя из следующего математического определения:*

$$x \,\vdots\, y \;\Leftrightarrow\; \exists\, t \leqslant x \; (x = y \cdot t).$$

Использовать рекурсию или ко-рекурсию.

Альтернативное определение для частного, остатка и делимости

Исходя из соотношения:

$$[x/y] = \sum_{i=1}^{x} \mathrm{sg}'(iy \mathbin{\dot-} x),$$

написать альтернативное определение частного и остатка при деление двух натуральных чисел, а также для вычисления $(x \,\vdots\, y)$, используя:

$$x \,\vdots\, y \;\Leftrightarrow\; \mathrm{rest}(x, y) = 0.$$

Проверить различные ситуации с 0.

Функции частного и остатка для $\mathbb{Z}$

Усложним задачу. Рассмотрим теперь задачу нахождения частного и остатка на все целые числа $\mathbb{Z}$. Все функции Haskell: **`quot`**, **`rem`**, **`div`**, **`mod`**; и пара функций, сразу возвращающих частное остаток: **`quotRem`** и **`divMod`** не соответствуют следующему математическому определению (проверить!).

Определение 1. Будем говорить, что целое число a делится с остатком на ненулевое целое число b, если

$$\exists q \in \mathbb{Z}\, \exists r \in \mathbb{N} \quad (a = bq + r \;\;\&\;\; 0 \leqslant r < |b|).$$

Например, должны выполнятся следующие математические соотношения:

Пример 1.

$$13 = 3 \cdot 4 + 1, \qquad 13 = (-3) \cdot (-4) + 1,$$
$$(-13) = 3 \cdot (-5) + 2, \quad (-13) = (-3) \cdot 5 + 2.$$

Используя собственные реализации из предыдущих задач, написать функции, правильно вычисляющие частное и остаток для чисел **`Integer`**.

Дополнительные задачи с арифметикой

Будем использовать предыдущие функции. В этом разделе работаем с числами из $\mathbb{N}$. Для дальнейшей работы разрешается использовать стандартные функции вместо `pls`, `mns` и `mlt` для более эффективных вычислений.

Задача 7. *Написать функцию, вычисляющую число делителей числа* $\mathrm{nd}(x)$.

Задача 8. *Написать функцию, вычисляющую сумму делителей числа* $\mathrm{sumd}(x)$.

Задача 9. *Написать функцию-предикат, определяющую, простое или составное число, исходя из математического определения:*

$$\mathrm{Prime}(x) \;\Leftrightarrow\; (x \geqslant 2)\ \&\ \forall t \leqslant x\, (x \,\vdots\, t \to (t = 1 \vee t = x)).$$

Задача 10. *Написать функцию, вычисляющую число простых делителей числа* $\mathrm{pnd}(x)$.

Задача 11. *Написать функцию* $\text{НОД}(x, y)$.

Задача 12. *Написать функцию* $\text{НОК}(x, y)$.

Оператор минимизации и его использование

В теории алгоритмов есть определение следующего оператора минимизации $M(g)$:

$$f(\overline{x}) = M(g)(\overline{x}) = \begin{cases} y, & \text{если } \big((\forall t \leqslant (y-1)\;\; g(t,\overline{x})\downarrow \neq 0) \\ & \text{ и } (g(y,\overline{x})\downarrow = 0)\big); \\ \uparrow, & \text{иначе.} \end{cases}$$

Здесь строка с «↑» означает, что в этом случае функция не определена, а $g(y, \overline{x}) \downarrow = 0$ означает, что для данных значений y и $\overline{x}$ значение функции определено и равно 0. Выражение $\overline{x}$ означает как обычно в математике возможность нескольких аргументов, например, $(x_1, \dots, x_n)$, но нам в дальнейшем будет достаточно работы работы с одним аргументом x и с двумя аргументами x_1 и x_2.

Таким образом, M — это математический оператор, который принимает на вход функцию $g(t, \overline{x})$, возвращает новую функцию $f(\overline{x})$ (g и f задаются на натуральных числах $\mathbb{N}$, $f = M(g)$ — это новая функция, которая применяется к аргументам $\overline{x}$). Для конкретного набора $(\overline{x})$ он находит, если возможно, минимальное значение y для которого $g(y, \overline{x}) = 0$.

Задача 13. *Для случая одного аргумента $g(t, x)$ (или, потом, для двух аргументов $g(t, x_1, x_2)$) реализовать оператор на Haskell, используя ко-рекурсию.*

Иногда рассматриваются варианты этого оператора: напр., ограниченный оператор минимизации, когда еще передается число или функция $\alpha(\overline{x})$, вычисляющая ограничения шагов перебора; или оператор минимизации, когда вместо функции $g(t, x)$ передается предикат $R(t, x)$, возвращающий истину или ложь.

Задача 14. *Для предиката $R(t, x)$ реализовать оператор на Haskell, используя ко-рекурсию.*

Задача 15. *Используя полученные функции на Haskell, создать функцию $[\sqrt{x}]$:*

$$[\sqrt{x}] = M(x < (t+1)^2)(x).$$

Задача 16. *Используя полученные функции на Haskell, создать функцию $[x_1/x_2]$:*

$$[x_1/x_2] = M(x_1 < (t+1)x_2)(x_1, x_2).$$