Контрольная по функциональному программированию

Задание 3

Весенний семестр 2021 г.

Известно, что ряд

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots$$

сходится к $\pi/4$. С помощью `scanl1` напишите функцию `approxPi :: [Double]`, представляющую бесконечную последовательность частичных сумм этого ряда, умноженных на 4.

Ряд выше сходится достаточно медленно. Однако существует так называемое преобразование Эйлера, ускоряющее сходимость знакопеременных рядов. Оно отображает последовательность $\{a_n\}_{n=0}^{\infty}$ в $\{b_n\}_{n=0}^{\infty}$, так что $\displaystyle\sum_{n=0}^{\infty}(-1)^n a_n = \sum_{n=0}^{\infty}(-1)^n b_n$, но второй ряд сходится быстрее. Обратите внимание, что преобразование принимает и возвращает последовательность членов ряда без знака.

Оператор конечной разности 1-го порядка определяется следующим образом.

$$(\Delta a)_k = a_{k+1} - a_k$$

Оператор конечной разности n-го порядка является n-кратной композицией Δ, то есть $\Delta^n = \Delta \circ \ldots \circ \Delta$ (n раз). Преобразование Эйлера отображает $\{a_n\}_{n=0}^{\infty}$ в $\left\{\dfrac{(\Delta^n a)_0}{2^{n+1}}\right\}_{n=0}^{\infty}$.

Напишите функцию `delta :: [Double] -> [Double]`, реализующую оператор конечной разности 1-го порядка. С помощью функций `iterate` или `unfoldr` из `Data.List` напишите функцию `euler :: [Double] -> [Double]`, осуществляющую преобразование Эйлера. Вычисление последовательности $\Delta^{n+1}a$ и коэффициента 2^{n+1} должны использовать ранее найденные $\Delta^n a$ и 2^n; эти выражения не должен вычисляться заново для каждого нового n. Наконец, напишите функцию `fastApproxPi :: [Double]`, представляющую бесконечную последовательность частичных сумм преобразованного знакопеременного ряда, умноженных на 4. Проверьте скорость схождения обоих рядов к π.

Функции в этом задании не должны явно использовать рекурсию.