Haskell как первый язык программирования

Абрамов С. М.

Пятнадцатая конференция «Свободное программное обеспечение в высшей школе»

07 февраля 2020 Переславль-Залесский

История курса

- **2005 год**, Университет города Переславля имени А.К.Айламазяна
- **Авторы:** Абрамов С. М., Пармёнова Л. В., Юмагужин Н. В.

Чтение курса:

- 2005–2017 — УГП имени А.К.Айламазяна, **обычно 3–5 курсы, 2 семестра, 60 пар**
- 2018–2019 — средняя школа № 2101 г.Москвы, **10–11 классы, 3 семестра, 45 пар**
- Осень 2019 — ЯрГУ имени П.Г. Демидова, **4 курс бакалавриата, 1 семестр, 15 пар**
- Осень 2019 — филиал МГУ имени М.В. Ломоносова в г.Севастополе, **1 курс бакалавриата, 1 семестр, 30 пар, полностью дистанционно**

Материалы по курсу

- https://goo.gl/r2IKNz

Прохождение курса

- Посещаемость
- Активность у доски
- Задачи
- Контрольные мероприятия (вопросы)
  - Рубежный контроль, зачеты, экзамены

Почему Haskell?

Быстрый выход новобранца на написание достаточно серьезных задач за счет свойств языка:

- Строгая статическая типизация
- Чистая функциональность
- Ленивая семантика (в т.ч. бесконечные значения)
- Параметрический полиморфизм, классы
- Функции высшего порядка
- Освобождение программиста от управления памятью (захват, освобождение, сборка мусора)

Все это позволяет концентрироваться на алгоритмической сути задачи, описывать решение в декларативном стиле

Цель курса, охват тем

Цель курса: знакомство с функциональной парадигмой программирования и развитие алгоритмических навыков

Охват тем:

- **История языков программирования** крупными мазками
- **Haskell** (за исключением монад) и **функциональная парадигма**:
  - **функции**, **вычисление** выражения через **редукции**, свойство Черча-Россера, **ленивые** и **энергичные** вычисления, функции **высшего порядка**, частичная и полная параметризация, частичное связывание, **λ-выражения**
  - **типизация** (слабая и сильная) и контроль типов (строгий статический—динамический)
  - **параметрический полиморфизм**
  - **классы** (наследование, переопределение)
  - **определение новых типов** и **новых классов**, **АТД**, **модули**
- Системы программирования Haskell (**HUGS**)
- **Алгоритмы и структуры данных**

Алгоритмы и структуры данных

- **Целочисленные задачи:** делимость, простые числа, Григорианский календарь, пифагоровы треугольники
- **Вычислительные задачи:** итерационная формула Герона, численное дифференцирование, метод Ньютона (нули функции), обратная функция
- **Списки:**
  - **Сортировки:** быстрая сортировка, сортировка вставкой, сортировка слиянием
  - **Функции высшего порядка:** filter, map, fold{l,r}, zip{With}, scan{l,r}, …
  - **Абстракции списков** (в т.ч. задачки от А.В. Шкреда)
  - **Бесконечные списки:** список всех простых чисел, список чисел Фибоначчи, генератор псевдослучайных чисел
- **Деревья и деревья поиска**, операции с деревьями, сортировка списка через построение дерева поиска
- **Вектора и матрицы**, операции с ними

«Доброволец — к доске!»

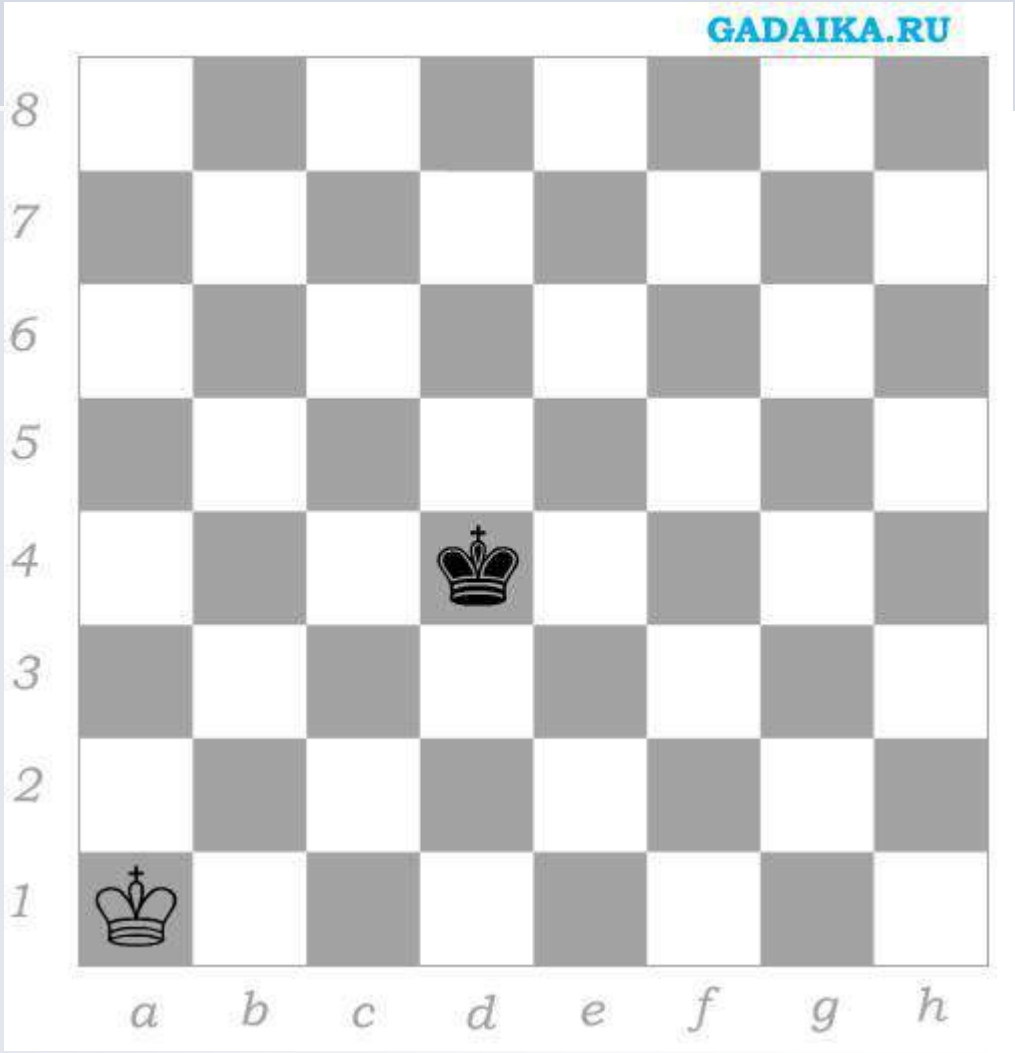
Слайды содержат включения работы студента у доски (много таких фрагментов создано на базе задачек из *Facebook-a* А.В. Шкреда):

- Излагается порция материала
- Ставится задача для программирования
- Звучит команда «Доброволец — к доске»
- Студент у доски (зал помогает) пишет код решения
- После полировки студент садится, лектор переходит к следующему слайду
- Сравниваем решения на доске и на слайдах
- Запускаем код в HUGS. Смотрим на поведение алгоритма. Обсуждаем улучшения

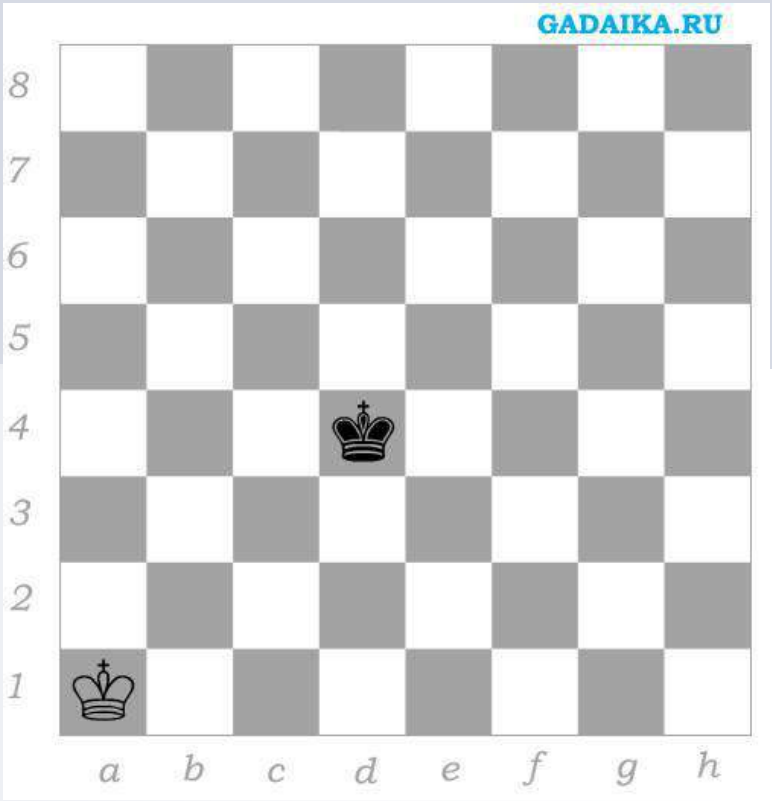
Задача А. Шкреда
Шахматы

2015.11.09 17:57

Поставьте 2 белые ладьи и коня так, чтобы возник мат черному королю. Мат должен именно сразу возникнуть, ходов делать нельзя, просто мат в 0 ходов



Решение




```
-- позиции королей
bk, wk :: (Int,Int)

bk          = (4,4)
wk          = (1,1)
```

```
-- вся доска и поля под ударами королей
cb, bitsBk, bitsWk :: [ (Int,Int) ]

cb          = [ (x,y) | x<-[1..8], y<-[1..8] ]
bitsBk      = [ (x,y) | x<-[3..5], y<-[3..5] ]
bitsWk      = [ (x,y) | x<-[1..2], y<-[1..2] ]
```

Решение

```
-- (x,y) на доске
inCb :: (Int,Int) -> Bool
inCb (x,y) = 1 <= x && x <= 8 && 1 <= y && y <= 8

-- бой ладьи
bitsRook :: (Int,Int) -> [ (Int,Int) ]
bitsRook (x,y) = [ (x,y') | y' <- [1..8]] ++ [ (x',y) | x' <- [1..8]]

-- бой коня
bitsHorse :: (Int,Int) -> [ (Int,Int) ]
bitsHorse (x,y) =  filter inCb ([ (x+d1,y+d2) | (d1,d2)<-hs ])
                   where  hs = [ (1,2), (2,1), (1,-2), (2,-1),
                                 (-1,2),(-2,1),(-1,-2),(-2,-1) ]

-- безопасная позиция p белой фигуры. bits — прикрытие других
isSave :: (Int,Int) -> [(Int,Int)] -> Bool
isSave p bits = (p `notElem` bitsBk) || (p `elem` bits)
```

Решение

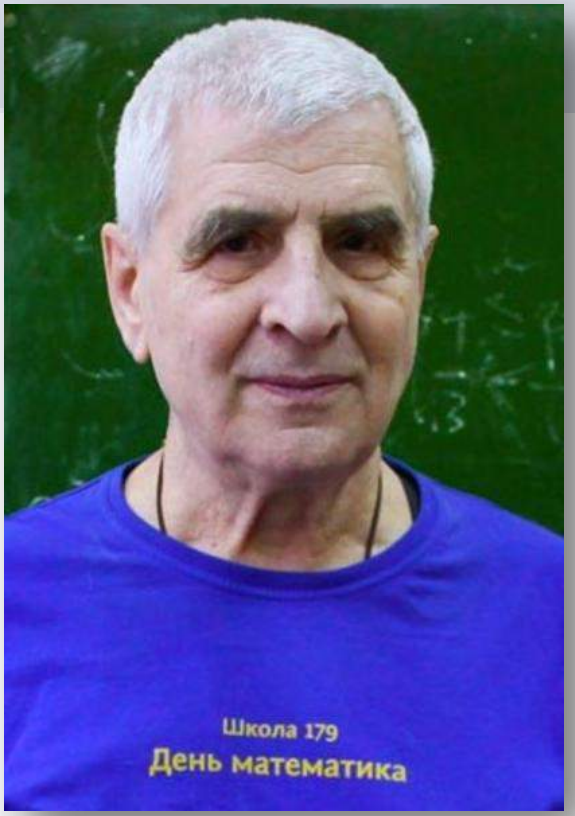
```
-- решение задачи (в координатах [1..8]x[1..8])
ps :: [[(Int, Int)]] -- [ [r1,r2,h] ]
ps =    let cb1 = delete bk (delete wk cb)
        in [  [r1,r2,h]  | r1 <- cb1, r2 <- cb1, r1 < r2,
                           h <- (delete r1 (delete r2 cb1)),
                           let br1 = bitsRook r1,
                           let br2 = bitsRook r2,
                           let bh  = bitsHorse h,
                           -- черный король и достижимые поля --- под боем
                           all (`elem` (br1++br2++bh)) bitsBk,
                           -- наши фигуры в безопасности
                           isSave r1 (br2++bh++bitsWk),
                           isSave r2 (br1++bh++bitsWk),
                           isSave h  (br1++br2++bitsWk)  ]

-- перевод в шахматную нотацию
ps' =   map (map f) ps
        where f (x,y) = ("+abcdefgh"!!x):("+12345678"!!y):[]
```

Материалы курса

- **370 слайдов лекций основного курса с работой у доски** — семестр, 30 лекционных пар (45 астрономических часов)
- **317 слайдов дополнительного курса Paul Hudak «The Haskell School of Expression»** — семестр, 30 лекционных пар (45 астрономических часов)
- **120 вопросов** для письменного минимального контроля — random-билеты равной сложности по 12 вопросов на 45 минут письменной работы
- **112 задач для самостоятельного программирования** — разной сложности
  - Сервер «автопроверки» программ студента

Самостоятельное программирование

Идеал? —
«Листочки Ник. Ник. Константинова»



- **Изучение материала задачи**
  - **Право на консультацию:** «не понимаю, как подойти к решению»
- **Написание кода и локальная отладка**
- **Сервер автопроверки** http://haskell.pereslavl.ru:
  - Test ... passed successfully
  - Test ... failed on: … bad result: … expected: …
- **Проверка кода преподавателем:** качество стиля написания, чистота и красота кода
  - «Ошибки стиля.docx»

Ошибки стиля и улучшение стиля

Странный стиль	Пишут так
F P = if C then True else False	F P = C

F P
  | C = True
  | otherwise = False
	F P = C

F P = if C then True else E	F P = C || E

F P
  | C = True
  | otherwise = E
	F P = C || E

F P = if C then False else E	F P = (not C) && E

F P
  | C = False
  | otherwise = E
	F P = (not C) && E

F P = if C then E else False	F P = C && E

F P
  | C = E
  | otherwise = False
	F P = C && E

and [a `mod` c == 0, b `mod` c == 0]	a`mod`c==0 && b`mod`c ==0
sum [product[b, b], product [(-4), a, c]]	b*b - 4*a*c
[ ]++XS	XS
[X]++XS	X:XS
[X]++[Y]++YS [X,Y]++YS	X:Y:YS

Ошибки стиля и улучшение стиля

Странный стиль	Пишут так
concat[[X], [Y], YS]	X:Y:YS
concat[X, Y ,Z]	X++Y++Z
foldl (+) 0 или foldr (+) 0	sum
foldl (*) 1 или foldr (*) 1	product
any (==x) xs или and (map (==x) xs)	x `elem` xs
filter (==x) ys == []	x `notElem` xs
f ns ms | null ns || null ms = E | …head ns … head ms… = …head ns … head ms… …tail ns … tail ms… | …head ns … head ms… = …head ns … head ms… …tail ns … tail ms…	f [ ] ms = E f ns [ ] = E f (n:ns) (m:ms) | …n… m… = …n… m… …ns…ms… | …n… m… = …n… m… …ns…ms…
f m n | aa(bb(cc(dd n)))> aa(bb(cc(dd m))) =n | otherwise =m	f m n | p n > p m =n | otherwise =m where p x = aa(bb(cc(dd x)))
inRng a b xs = filter (pr a b) xs where pr a b x =(x>=a)&&(x<=b)	inRng a b xs = filter (pr) xs where pr x = (x>=a)&&(x<=b)

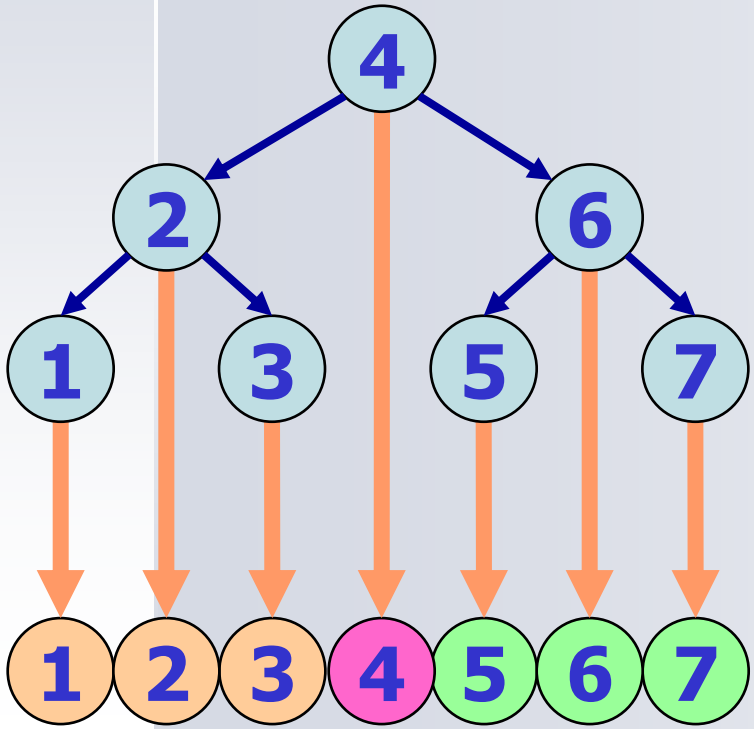
Самостоятельное программирование

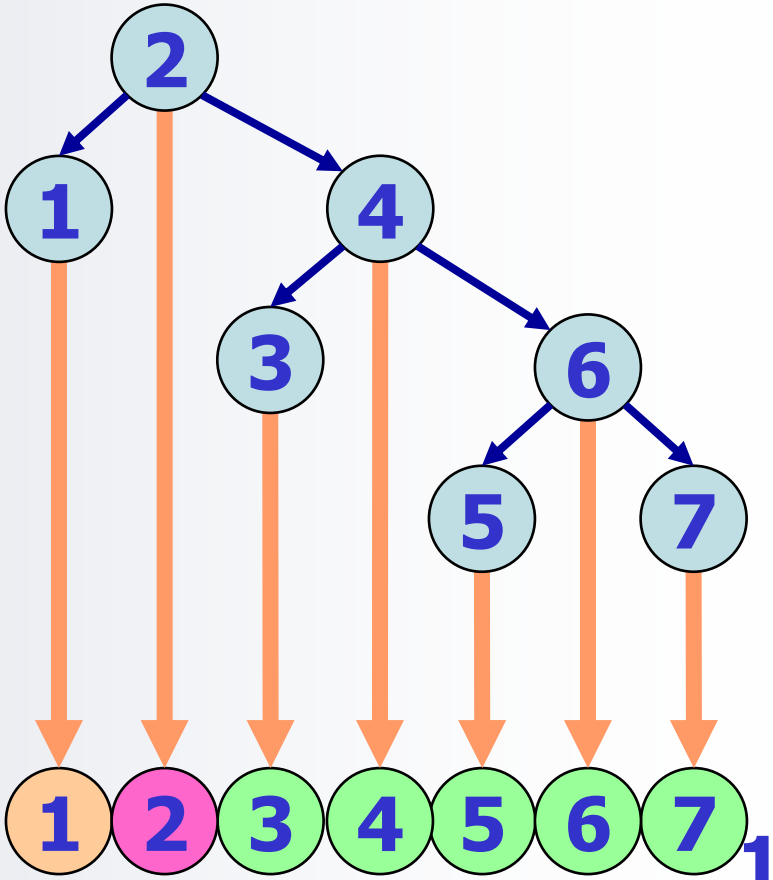
112 задач разной сложности и на тренировку разных навыков

- **Разминка** — задачки, которые новичок решит в одну строку
- **Десяток несложных однотипных задач вида** «найти минимальный корень уравнения ...» и «найти все (список) корней уравнения ...» — этюд на навык выделения фрагментов со свойством «reusable code»
- Тот же навык закрепляется в этюдах на различные **позиционные системы счисления** (включая «Сетунь»)
- **Вполне серьезные задачи**
  - Для двух разных представлений **полиномов одной переменной написать «полный комплект» операций**: упрощение полиномов, сравнение на равенство, сложение, вычитание, возведение в степень, деление с остатком, дифференцирование, вычисление для заданного значения переменной
  - Примитивы работы с **формулами языка логики высказываний** и с **арифметическими выражениями с переменными**

Пример задач

```
data Tree a = Leaf | Node a (Tree a) (Tree a)
-- flatten  :: Tree a -> [a]
trees       :: [a] -> [ Tree a ]
```





Пример задач

```
split :: [a] -> [([a],a,[a])]
split [x]     = [ ([], x, []) ]
split (x:xs) = ([], x, xs) : [ (x:lft, a, rgt) | (lft, a, rgt) <- split xs ]

Main> split [1,2,3,4]
[([],1,[2,3,4]),([1],2,[3,4]),([1,2],3,[4]),([1,2,3],4,[])]

trees :: [a] -> [ Tree a ]
trees [ ] = [ Leaf ]
trees xs = [ Node a tL tR | (lft, a, rgt) <- split xs,
                            tL <- trees lft,
                            tR <- trees rgt
           ]

Main> trees [1,2,3]
[Node 1 Leaf (Node 2 Leaf (Node 3 Leaf Leaf)),
  Node 1 Leaf (Node 3 (Node 2 Leaf Leaf) Leaf),
  Node 2 (Node 1 Leaf Leaf) (Node 3 Leaf Leaf),
  Node 3 (Node 1 Leaf (Node 2 Leaf Leaf)) Leaf,
  Node 3 (Node 2 (Node 1 Leaf Leaf) Leaf) Leaf]
```

Пример задач
Числа Каталана

```
trees :: [a] -> [ Tree a ]
trees [ ] = [ Leaf ]
trees xs = [ Node a tL tR | (lft, a, rgt) <- split xs,
                            tL <- trees lft,
                            tR <- trees rgt
           ]

Main> length (trees [1..10])
16796

nTrees :: Int -> Integer
nTrees 0 = 1
nTrees n = sum [ (nTrees llft)*(nTrees lrgt) | llft <-[0.. (n-1)],
                                               let lrgt = n-1-llft ]

Main> nTrees 10
16796
```

Пример задач
Числа Каталана

```
nTrees :: Int -> Integer
nTrees 0 = 1
nTrees n = sum [ (nTrees llft)*(nTrees lrgt) | llft <-[0.. (n-1)],
                                               let lrgt = n-1-llft ]
```

```
Main> nTrees 14
2674440
(184,144,309 reductions, 265,455,104 cells, 26 garbage collections)
```

```
nTrees' :: Int -> Integer
nTrees' n = nts!!n

nts :: [Integer]
nts = 1 : [ f n | n<-[1..] ]
        where f n = sum [ (nts!!llft)*(nts!!lrgt) | llft <-[0.. (n-1)],
                                                    let lrgt = n-1-llft]
```

```
Main> nTrees' 14
2674440
(23,861 reductions, 31,239 cells)
```

Speed up factor ≈7,000

Пример задач
Числа Каталана

```
nTrees' :: Int -> Integer
nTrees' n = nts!!n
nts :: [Integer]
nts = 1 : [ f n | n<-[1..] ]
        where f n = sum [ (nts!!llft)*(nts!!lrgt) | llft <-[0.. (n-1)],
                                                    let lrgt = n-1-llft]
```

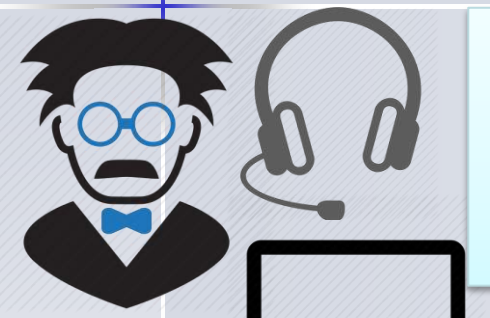
```
Main> nTrees' 100
896519947090131496687170070074100632420837521538745909320
(6,607,376 reductions, 8,471,665 cells)
```

Speed up factor
≈54
≈420,000

```
nTrees'' :: Int -> Integer
nTrees'' n = nts'!!n
nts' :: [Integer]
nts' = 1 : [ sum (zipWith (*) ns rs) | n<-[1..], let ns = take n nts',
                                       let rs = reverse ns ]
```

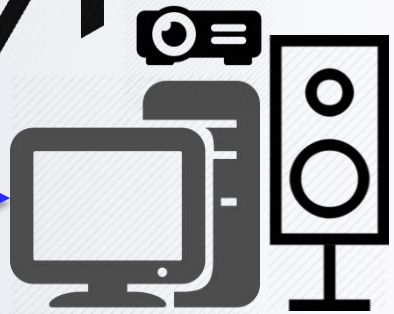
```
Main> nTrees'' 100
896519947090131496687170070074100632420837521538745909320
(121,876 reductions, 282,815 cells)
```

Организация дистанционного курса



Skype-сеанс:
«демонстрация экрана» преподавателя
+ звук в обе стороны

Компьютер со Skype-ом
+ колонки
+ радиомикрофон
+ проектор
+ экран



Консультации?
WatsApp-группа.

Email +
Dropbox для сдачи работ

Работа у доски: google-документ
совместное редактирование

«Доброволец»
минимальный компьютер
+ Web-браузер
+ мышь
+ клавиатура
+ экран

Планы и мечты по дальнейшему развитию

- Развитие сервера автопроверки — здесь ясно, что надо доработать;
- Расширение и улучшение набора задач — полезно взаимодействие с любыми заинтересованными партнерами.
- Идеал? — Листочки Н.Н.Константинова
- Перевод курса в статус «открытого проекта» — нужны партнеры, обладающие технологическими навыками

Haskell

A Purely Functional Language

Спасибо за внимание! Готов ответить на вопросы!

Абрамов Сергей Михайлович

abram@botik.ru