**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”**

**Факультет** Программной инженерии и компьютерной техники

**Направление подготовки (специальность)** Системное и прикладное программное обеспечение

### **ЭСсе** на тему:

# Курс “Функциональное программирование на языке Haskell”

Обучающийся Орловский М.Ю P4116

(Фамилия И.О.) (номер группы)

Санкт-Петербург

2023 г.

Курс является записью лекций от Computer Science Center, видимо и проводился в том числе на его базе. Это значит, что вероятно, это один из лучших курсов по Хаскелю на русском языке. Я склоняюсь к тому, что это правда, однако следует также в дальнейшем ознакомится со вторым курсом (от того же преподавателя) чтобы лучше перенять «программирование от типов» которое имманентно языку.

Раз уж я затронул эту тему, расскажу немного о Хаскеле. Это функциональный язык программирования, основными чертами которого являются ленивость и очень сильная система типов. Такая система типов является одной из самых сильных из тех, что я встречал (сильнее только, наверное, у пруверов, напр. Coq). Она не допускает сайд-эффекты в функции, если в её сигнатуре не обозначено иное. Это значит, что необходимо делать операции ввода-вывода, изменение состояния (переменных как концепции в языке нет), доступ к внешнему миру и т. д. посредством специальных типов. Точнее сказать, тайпклассов (грубо говоря интерфейсов), которые имеют специфические правила реализации. Одним таким ходовым (и мемным) тайпклассом можно назвать монаду (Monad), которая представляет собой в общем случае «контекст вычислений». Он настолько ходовой, что в Хаскель присутствует специальное ключевое слово do, которое перемещает процесс вычисления в контекст определенной монады. Также, стоит вспомнить моноид и функтор, которые тоже используются активно, в т. ч. для реализации монады (монада — это моноид в категории эндофункторов).

Также, в языке присутствует параметрический полиморфизм, что позволяет реализовывать функции, подходящие для определенного набора типов (или даже для любых типов). Например, функция, возвращающая свой аргумент будет выглядеть так:

id :: a -> a

id x = x

Она будет работать на любом типе, будь то примитивный тип или определенный пользовательский.

Можно еще много расписать о серьезных, с теоретической точки зрения, фичах языка, но я остановлюсь на том, что я действительно использовал (или буду использовать) в дальнейшем:

* Я реализовал вывод типов алгоритмом Хиндли-Милнера в компиляторе в ходе работы над курсом «Разработка компилятора», хотя он и не поддерживает, например, рекурсивные функции (из-за особенностей let-полиморфизма, алгоритм сильно меняется)
* Очень хорошо на примере функционального языка было объяснено, что такое система типов, почему она такая и зачем она нужна, что также помогло мне как с изучением статического анализа, так и подтолкнуло меня на выбор курса «Системы типизации лямбда-исчисления»
* Сам язык я наверняка не буду использовать (не в обозримом будущем), но идеи «чистая функция», «сайд-эффект», «конструктор типов» и другие, мною уже неоднократно встречались и будут встречаться в дальнейшем, в ходе моего обучения концепциям языков программирования
* Хаскель также является примером формально верифицируемого языка, что помогло мне оценить строгость семантического анализа, который в нем проводится и ввело для меня понятия «soundness» (непротиворечивость) и «completeness» (полнота)

Также, прикладываю сертификат о выполнении

A screenshot of a computer

Description automatically generated with low confidence