Полозов Александр, 2010

Курс «Функциональное программирование и основы теории Computer Science»

Цели курса

- Продемонстрировать непривычный подход к решению задач разработки программных систем, отличный от традиционно преподаваемого в учебных заведениях, однако набирающий популярность в индустрии Software Development.
- Привить умение анализировать поставленную задачу, выделять в ней абстракцию, производить разбивку на подзадачи. Дать понять преимущества манипулированиями абстракциями предметной области в коде, в отличие от низкоуровневых команд языка программирования.
- Научить использовать полезные идиомы, свойственные функциональному подходу, вне зависимости от используемого языка и решаемой задачи.
- Показать практическое насущное применение ФП. В числе примеров:
  - Алгоритмы параллельного и асинхронного программирования, которые сейчас являются одной из наиболее активно исследуемых областей в IT. В курсе им уделено особое внимание.
  - Принципы работы с объемными потоками/списками данных.
  - Парсинг текста, создание DSL (Domain Specific Language).
- Познакомить с некоторыми математическими основаниями, на которых строится предмет Computer Science. Показать, что знания теории алгоритмов действительно применимы в практической разработке.

Предварительные знания

1. Программирование и алгоритмические языки: необходимо свободно обращаться хотя бы с одним алгоритмическим языком программирования, вероятнее всего – императивным, использовать его для реализации алгоритмов и структур данных, уметь придумывать несложные алгоритмы. Желательно знакомство с объектно-ориентированной парадигмой.
2. Дискретная математика: теория множеств, графы, отношения, абстрактная алгебра в пределах операций с полугруппами и моноидами.
3. Математическая логика и теория алгоритмов: знакомство с машиной Тьюринга, по возможности с другими вычислительными моделями; тезис Черча-Тьюринга.

Организация

Курс рассчитан на 1 семестр, с длиной лекции в 1 пару (2 академических часа). Исключение составляет вводная лекция, которая совмещена с первой парой и дает суммарную длину занятия в 3 академических часа.

Большая часть практических примеров отведена на самостоятельную работу и выносится в домашнее задание. Проверка и оценивание домашних заданий, выполненных студентами, выполняется в online-режиме в течение рабочей недели с дня лекции. Тем не менее, по просьбе студентов возможно включение в план 1-2 дополнительных практических занятий.

Программа курса

I. Вводная лекция. Функциональный подход к программированию. Применения.

II. Идиомы функционального программирования.

1. Рекурсивные и итеративные процессы, их отличие и особенности. Вычисление с аккумулятором. Хвостовая рекурсия. Процесс декомпозиции задачи. Язык программирования F#.
2. Функция как объект манипулирования. Функции высших порядков. Абстракция вычисления. Часто используемые операторы над функциями. Каррирование.
3. Замыкания. Проблемы реализации замыканий, эмуляция в императивных языках. Работа со списками, списочные комбинаторы. Пример: LINQ.
4. Абстракция данных. Алгеибраические типы данных. Сопоставление с образцом. Список как алгеибраический тип данных. Деревья и операции над ними. Примитивный синтаксический разбор.

III. Гомоморфизмы и параллельное программирование.

5. Списочная свертка и её применения. Continuation-passing style. Инструкция `call/cc`. Трансформация рекурсии в цикл.
6. Моноиды. Списочные гомоморфизмы. Третья теорема о гомоморфизмах. Сбалансированные деревья и дерево отрезков. Сканирующие пробеги и их применения в параллельном программировании. Сегментированные пробеги. Примеры: алгоритмы параллельного программирования по данным, использующие идиому пробега – выпуклая оболочка, максимальный поток в графе, численные методы. Технология Google MapReduce и ее открытая реализация Apache Hadoop. Примеры MapReduce: индекс цитирования Web, Байесовские классификаторы, пути в графе.
7. Обобщение свертки на алгебраические типы данных. Свертка деревьев. Понятие катаморфизма.

IV. Сочетание различных подходов.

8. Математические основы функционального программирования. Бестиповое λ-исчисление. Порядок редукций и теорема Черча-Россера. Теорема о неподвижной точке. Y-комбинатор. Пример использования: кеширование ответов функции в динамическом программировании «сверху вниз».
9. Концепция изменяемого состояния. Проблемы, связанные с присваиванием и способы избавления от них в практической разработке. Чисто функциональные структуры данных. Понятие потока и его связь со списками. Ленивые вычисления. Язык программирования Haskell.
10. Проблема побочных эффектов. Монада как абстракция последовательного вычисления. Примеры: монады IO, List, Maybe, State, Parser. F# computational workflows. Пример: использование библиотек Parsec/FParsec для реализации DSL, профессиональный синтаксический разбор. Асинхронные вычисления.

V. Основы типизации. Обзор систем типов и алгоритмов вывода типов, особенности типизированного λ-исчисления. Разговор о возможностях, теоретически доступных алгоритмическим языкам. Теорема Райса.