ФУНКЦИОНАЛЬНОЕ ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Санкт-петербургский государственный политехнический университет
Институт Компьютерных Наук и Технологий
Кафедра: Информационные и Управляющий Системы
Автор: Лукашин Антон Андреевич

2016-2017

Содержание

- Коллекции
  - *List, Map, Set, Range*
  - *Mutable/immutable*
  - *Java collections*
- Циклы
  - *while*
  - *for*
  - *foreach*
- Lambda-операции
  - *map, flatMap, flatten*
  - *reduce, reduceLeft, reduceRight*
  - *filter, find*
  - *sort*
  - *foldLeft, foldRight*
- Pattern matching

Pattern matching

- Обобщение switch в C++\Java\.net
- Синтаксис
  - *Выражение*
  - *match*
  - *Набор case сопоставляемых выражениям*
  - *MatchError exception*
  - *Pattern (шаблоны)*
    - Конструктор
    - Переменная
    - Wildcards _
    - константы
- Пример
  - *val result = “Hello” match {case String(s)=>println(s+” World”)}*

Pattern matching

- Конструктор
  - *Сопоставляет тип (или подтип), который создается конструктором с указанными параметрами*
- Шаблон с переменной сопоставляется переменной подходящего типа, которая получает имя этой переменной
- Шаблон с константой сопоставляет выражение с этой константой (==)

List

- List
  - *Фундаментальное представление данных в ФП*
  - *Список x1, x2…xn –> List(x1, x2…xn)*
  - *val nums = List(1,5,15)*
- Отличительные особенности
  - *Immutable (!!!)*
  - *Работа со списками рекурсивна, тогда как с массивами прямая*
  - *Все элемент списка одного типа (List[Nothing] = List())*
  - *Правая ассоциативность (A :: B :: C === A :: (B :: C))*
- Построение списков
  - *Nil – пустой список*
  - *:: - оператор констуркции списка (x :: xs)*
- Операции над списками
  - *head*
  - *Tail*
  - *isEmpty*

List

- Больше методов
  - *length*
  - *last*
  - *Init (xs :except last)*
  - *xs take n (первые n элементов или список целиком)*
  - *xs drop n (последние n элементов)*
  - *xs(n) || xs apply n*
  - *++ - concatenate two lists*
  - *reverse*
  - *updated (n,newElement)*
  - *indexOf*
  - *contains*

List и Pattern matching

- Nil сопоставляется пустому списку === List()
- l :: ls - сопоставляется элементу l и окончанию списка ls
- List(l1, l2, ,, ln) == l1 :: l2 :: ... :: ln :: Nil
- x :: y :: List(xs, ys) :: zs length?
  - *3*
  - *4*
  - *5*
  - >=*3*
  - >=*4*
  - >=5

List и Pattern matching

- Nil сопоставляется пустому списку === List()
- l :: ls - сопоставляется элементу l и окончанию списка ls
- List(l1, l2, ,, ln) == l1 :: l2 :: ... :: ln :: Nil
- x :: y :: List(xs, ys) :: zs length?
  - *3*
  - *4*
  - *5*
  - *>=3*
  - *>=4*
  - >=5

Implementations

- def last[T](xs:List[T]) : T = xs match {
  case List() =>
  case List(x) =>
  case y::ys =>

}

- def concat[T](xs:List[T], ys:List[T]) : List[T] = xs match {
  case List() =>
  case z::zs =>

}

Implementations

- def last[T](xs:List[T]) : T = xs match {
  case List() => throw smth
  case List(x) => x
  case y::ys => last(ys)

}

- def concat[T](xs:List[T], ys:List[T]) : List[T] = xs match {
  case List() => ys
  case z::zs =>z:: concat(zs, ys)

}

Функции высших порядков для коллекций

- map, flatMap
  - *Применяет операцию к каждому элементу списка*
  - *Раскрывает списки*
- filter, find
  - *Принимает предикат*
- Reductions
  - *reduceLeft((x,y)=>???)*
  - *foldLeft – аналогично reduceLeft + аккумулятор*
  - *reduceRight*
  - *foldRight*
- Реализация reverse с и без fold

Спасибо за внимание!

- Планы на следующую лекцию
  - *While loop*
  - *for loop*
  - *range*
  - *Stream*
- Самостоятельное изучение:
  - *generics*
  - *implicit*