Терория ФП

Лекция 5.

ВМК МГУ, 2010. Контакты: sakharov@lvk.cs.msu.su, http://haskell.secsem.ru

Императивные языки

- ASM инструкции языка – синонимы машинных команд.
- C/C++ – уровень абстракции выше, явная работа с памятью.
- Java/C# – байткод, виртуальная машина, сборка мусора.
- Python/Ruby/Lua – скриптовые языки

Проблемы императивных языков

Круг проблем, переживших «эволюцию» императивных языков программирования:

- Проблемы изменяемого состояния. (многотредовые приложения)
- Слабая выразительная мощь.

Функциональные языки

- Нет операции присваивания.
- «Ленивый» порядок редукций
- Программа – набор уравнений, а не описание последовательности действий.
- Строгая типизация

Базовые типы в Haskell

Базовыми типами в Haskell являются:

Char символы

Int целые числа

Integer целые числа неограниченного размера

Real вещественные числа

Bool булевы переменные

Списки в haskell

Список может содержать элементы только одного типа.

Определение пустого списка: []

Операция создания пары (конструктор): (:)

Получение головы списка: head

Получение хвоста списка: tail

Списки в haskell: примеры

Определение списка, содержащего набор элементов:
1:2:3:[]
Эквивалентное определение: [1,2,3]
Получение головы списка: head [1,2,3] – 1
Получение хвоста списка: tail [1,2,3] – [2,3]

Функции в haskell

Синтаксис описания функций в haskell:

<function name> <arguments> = <body>

Функции в haskell: примеры

Определение функции:

```
add x y = x + y
```

Вызов функции:

```
add 1 4
```

Функции в haskell: образцы

Образец – выражение, построенное с помощью операций конструирования данных, которое используется в функциях для сопоставления с данными.

Примеры:
6 – просто числовая константа
x – просто переменная
(x:y) – пара

Функции в haskell: клозы (1/1)

Вместо ветвления в теле функции можно использовать клозы.

Клоз – запись одного варианта вычисления, зависящего от вида образцов, записанных около имени функции.

Пример:

```
last [x] = x
last (x:t) = last t
```

Таким образом, определение функции – список клозов.

Функции в haskell: клозы (2/2)

Именованные образцы (@). К ним можно обращаться, как к целому, так и по частям.

Пример (дублирование головы списка):

```
headDup l@(x:t) = x:l
```

Образцы вида (n + k).

Пример (вычисление факториала):

```
fac 0 = 1
fac ( n + 1 ) = (n+1)*fac n
```

Кортежи в haskell

Кроме типа списков, которые могут содержать элементы одного типа, в языке haskell существует другой сложный тип данных – кортежи(tuples).

Кортеж содержит фиксированный набор элементов разных типов.

Элеметы заключаются в круглые скобки и разделяются запятыми.

Примеры:

```
(10, 'a' )
(10, (14, 'a','b') )
```

λ-абстракции в haskell

В Haskell λ-абстракции кодируются очень просто. Символ (λ) заменяется на символ (\), точки(.) заменяются на стрелку (->).

Пример:

```
λx.λy.(x+y)
\x -> \y -> x + y
```

(или в упрощенной форме:) `\x y -> x + y`

Модули в haskell

Haskell поддерживает модуляризацию программ. Модуль – повторно используемый компонент.

Именем модуля может быть любой идентификатор, начинающийся с заглавной буквы.

Определение модуля начинается служебным словом where, после которого указывается имя модуля. За именем модуля указывается список экспортируемых модулем сущностей и заканчивается служебным словом where.

```
modDecl ::= module <ModuleName> <ExportList> where <body>
```

Модули в haskell (пример)

```
module QSort where

q []     = []
q (x:t) = q [ y | y <- t, y < x ]
          ++ [x] ++
          q [ y | y <- t, y >= x ]
```

Модули в haskell (ghci)

Модули и интерпретатор языка haskell (ghci)

```
$ ghci ./QSort.hs
GHCi, version 6.8.2: http://www.haskell.org/ghc/  :? for
help
Loading package base ... linking ... done.
[1 of 1] Compiling QSort              ( ./QSort.hs,
interpreted )
Ok, modules loaded: QSort.
*QSort> q [4,5,6,7,1,2,3,4]
[1,2,3,4,4,5,6,7]
```