Лекция 2. Введение в Haskell User's Gofer System (hugs)

Пережогин А.С.

22 февраля 2012 г.

Интерпретатор hugs

- hugs – широко распространенный интерпретатор языка программирования Haskell 98
- hugs – интерпретатор функционального программирования, который удобно использовать для обучения
- Домашняя страница hugs находится по ссылке: www.haskell.org/hugs
- Начало разработки Haskell – 1987 год. Документация языка Haskell – 2003 год.

Запуск интерактивного режима hugs

В командной строке Linux набираем команду

```
[user@localhost]\% hugs
__   __ __  __  ____   ___     ------------------------------------
||   || ||  || ||  || ||__     Hugs 98: Based on the Haskell 98 standar
||___|| ||__|| ||__||  __||    Copyright (c) 1994-2005
||---||         ___||          World Wide Web: http://haskell.org/hugs
||   ||                        Bugs: http://hackage.haskell.org/trac/hu
||   || Version: September 2006_____________________________________

Haskell 98 mode: Restart with command line option -98 to enable extensi

Type :? for help
Hugs>|
```

Арифметические вычисления в hugs

Строка ввода команд в интерактивном режиме hugs позволяет выполнять арифметические операции

```
Hugs> 4 + 5
9
Hugs> 3*(4 + 5)
27
Hugs> -3*(4 - 5)
3
Hugs> (4+5)^2
81
Hugs> sqrt(3+6)
3.0
Hugs> 41/3
13.6666666666667
```

Стандартный файл Prelude.hs

Стандартные функции доступные в интерактивном режиме находятся в файле Prelude.hs. В дополнение к арифметическим операциям этот файл содержит функции для обработки списков.
Определение. Список в Haskell – это индексированная совокупность однородных значений, заключенных в квадратные скобки и разделенные запятой.

```
Hugs> [1,2]
[1,2] :: [Integer]
Hugs> [1,2,3.55]
[1.0,2.0,3.55] :: [Double]
```

Обзор функции работы со списками:

- head – возвращает первый элемент списка

  ```
  Hugs> head [6,7,1,8,12]
  6
  ```

- tail – удаление первого элемента из списка

  ```
  Hugs> tail [6,7,1,8,12]
  [7,1,8,12]
  ```

Стандартный файл Prelude.hs

- список !!*N* – выбор N-ого элемента списка (нумерация с 0)

```
Hugs> [6,7,1,8,12] !! 1
7
```

- take N список – выбор первых N элементов

```
Hugs> take 2 [6,7,1,8,12]
[6,7]
```

- drop N список – удаление N первых элементов из списка

```
Hugs> drop 2 [6,7,1,8,12]
[1,8,12]
```

- length список – число элементов списка

```
Hugs> length [6,7,1,8,12]
5
```

- sum список – сумма элементов списка

```
Hugs> sum [6,7,1,8,12]
34
```

- product список – сумма элементов списка

Стандартный файл Prelude.hs

```
Hugs> product [6,7,1,8,12]
4032
```

- список1 ++ список2 – объединение двух списков

```
Hugs> [6,7,1]++[8,12]
[6,7,1,8,12]
```

- reverse список – обращение порядка элементов в списке

```
Hugs> reverse [6,7,1,8,12]
[12,8,1,7,6]
```

Запись функций в Haskell

1. В математике принято обозначение функций
   $f(a, b) + cd - g(d)$ – вычисляется значение $f(.,.)$ при фиксированных a и b, далее прибавляем произведение c на d и вычитаем $g(.)$ в точке d
2. В Haskell применение функции к аргументам разделяется пробелами. Умножение обозначено $*$.
   $f\ a\ b + c * d - g\ d$

Функция имеет больший приоритет в выражениях. $f\ a + b$ – сначала вычисляется f a, затем прибавляется значение b.
Примеры:
$f(x)$ $f\ x$
$f(x, y)$ $f\ x\ y$
$f(g(x))$ $f\ (g\ x)$
$f(x, g(y))$ $f\ x\ (g\ y)$

Скрипт-файл в Haskell

1. Аналогично объявленным функциям в файле Prelude.hs программист самостоятельно может задавать свои функции
2. Новые функции определяются в отдельном скрипт-файле – текстовый файл, в котором задана последовательность определений функций
3. Для удобства скрипт-файл в Haskell имеет расширение .hs. Это условие не обязательно и определяется для удобства

Создание скрипт-файла

1. В командной строке Linux создадим файл
   touch 01first.hs
2. Установим редактор по умолчанию
   export EDITOR=vi
3. Запустим редактор hugs
4. Выполним команду
   Hugs> :load 01first.hs
5. Редактируем файл 01first.hs
   Main>:e
6. Клавиша i переводит редактор vi в режим вставки текста
7. Запишем строку в файл 01first.hs
   double :: Integer -> Integer
   double x = x*x
8. Выход из редактора ESC, Shift+Z,Z
9. Выполним в hugs
   Main> double 5
   25

Создание скрипт-файла. Второй способ

1. В командной строке Linux создадим файл
   touch 01first.hs
2. Редактируем файл 01first.hs
   vi 01first.hs
3. Клавиша i переводит редактор vi в режим вставки текста
4. Запишем строку в файл 01first.hs
   double :: Integer -> Integer
   double x = x*x
5. Выход из редактора ESC, Shift+Z,Z
6. Запускаем hugs
   hugs 01first.hs
7. Main> double 5
   25
8. В новом окне открываем для редактирования файл 01first.hs.
   Добавляем новую строку
   factorial n = product [1..n]

Создание скрипт-файла. Второй способ

9. В hugs даем команду :reload, чтобы интерпретатор перезагрузил изменения файла 01first.hs
10. Main>factorial 5
    120

Требования на имена функций и переменных

1. Функции и аргументы должны начинаться со строчных букв
   `myNew, func1, argument_1, arg_2`
2. Соглашение для переменных хранящих списки. В имя переменной добавляется окончание s.
   xs, ns, nss

Правило выравнивания

Во избежании набора открывающих и закрывающих скобок используется правило выравнивания текста.
Элементы расположенные в одном столбце относятся к одной и тоже открывающейся скобке.
Правильно:

$$a = 10$$
$$b = 7$$
$$c = 4$$

Неправильно:

$$a = 10$$
$$\quad b = 7$$
$$c = 4$$

Правило выравнивания. Пример Haskell

Код программы с правилом выравнивания

$a = b + c$
  where
    $b = 1$
    $c = 2$
$d = a * 2$

эквивалентен

$a = b + c$
  where
    $\{b = 1;$
    $c = 2\}$
$d = a * 2$

Команды интерпретатора hugs

- :load name.hs – загрузка скрипта name.hs
- :reload – перезагрузка текущего файла
- :edit name.hs – редактирование файла name.hs
- :edit – редактирование текущего файла
- :type expression – отображение типа выражения expression
- :? – список доступных команд
- :quit – выход из hugs

Контрольные задания

- Создать файл 02train.hs
- Написать функцию cube, которая возводить в третью степень целое число
- Проверить работу функции cube в hugs
- Реализовать функцию $y = 2x + 3$ в 02train.hs
- Проверить работу функции y в hugs
- Создать список m степеней числа 2, начиная с 1 до 5 степени включительно в файле 03list.hs
- Вывести первый элемент списка m
- Вывести список всех элементов, кроме первого
- Посчитать число элементов списка m
- Создать 2 списка: m1=[1,2] и m2=[2,3]. Объединить списки m1 и m2.