Лекция 12. Ленивые вычисления в Haskell

Пережогин А.С.

9 октября 2012 г.

Введение

К настоящему моменту мы изучили как вычисляются выражения в Haskell. При вычислении мы используем простые принципы:

1. избегаем ненужные вычисления
2. делаем программу модульной
3. разрешаем программировать бесконечные списки

Техника вычислений называется ленивые (отложенные) вычисления и язык программирования Haskell называется ленивым функциональным языком.

Вычисление выражений

Выражения вычисляются или преобразуются по заданным определениям до тех пор, пока возможно упрощение. Пример. square n = n*n. Выражение square(3+4) может быть вычислено по следующей схеме

```
square(3+4) = square 7 = 7*7 = 49
square(3+4) = (3+4)*(3+4) = 7*(3+4) = 7*7 = 49
```

Оба способа вычисления одного и того же выражения в Haskell приводит к одному результату.

1. Внутренняя редукция
2. Внешняя редукция

Пример редукции

```
loop = tail loop
```

Вычислим выражение fst(1,loop).

1. Внутренняя редукция

   ```
   fst(1,loop) = fst(1, tail loop) = fst (1 , tail (tail loop)) =..
   ```

   бесконечное число шагов

2. Внешняя редукция

   ```
   fst(1,loop) = 1
   ```

   один шаг

Замечание: внешняя редукция требует большего числа операций, чем внутренняя.

Пример 2.

Проблема во внешней редукции решается с помощью указателей.

```
  square(3+4)
=
  (pointer * pointer)      pointer = (3+4)
=
  (pointer * pointer)      pointer = 7
=
  49
```

Ленивые вычисления = внешняя редукция + указатели

Бесконечный список

Определим список

```
ones :: [Int]
ones = 1 : ones
```

Получаем бесконечный список единиц

```
ones = 1:ones
     = 1:1:ones
     = 1:1:1:ones
     = ...
```

1. Внутренняя редукция

```
head ones = head (1:ones)
          = head (1:1:ones)
          = head (1:1:1:ones)
          =..
```

2. Внешняя редукция

```
head ones = head (1:ones)
          = 1
```

С помощью бесконечных списков можно генерировать конечные последовательности

```
> take 5 ones
[1,1,1,1,1]
> take 5 [1..]
[1,2,3,4,5]
```

Модульные вычисления: take 5 – управление, [1..] – данные.

Контрольные задания

- Создать программу генерации последовательности чисел Фибоначчи.
- Создать функцию вычисления n числа Фибоначчи.
- Создать последовательность простых чисел.