Лекция 9. Операции ввода/вывода

Пережогин А.С.

11 октября 2012 г.

Концепция чистого функционального языка программирования дает прекрасную возможность создать файл с описание функций и выполнить требуемую функцию с помощью интерпретатора. Однако, хотелось бы создать программу, которая имела бы возможность предоставлять интерактивный ввод с клавиатуры данных в программу.
Возникает ПРОБЛЕМА: функции в Haskell являются чисто математическими функциями. Программа на Haskell не имеет побочного влияния на ход выполнения. Но с другой стороны ввод значений с клавиатуры влияет на результат вычисления.
ПРОТИВОРЕЧИЕ.

Решение

В Haskell предопределен специальный параметризованный тип данных - тип операции ввода/вывода.

```
IO a
```

Например.

```
IO Char - тип действия возвращает значение Char
IO () - тип действия не возвращает ничего
() - кортеж без компонент
```

Базовые операции ввода/вывода

Основные 3 примитивные функции в базовой библиотеке Haskell

- getChar – считывает символ с клавиатуры

  ```
  getChar :: IO Char
  ```

- putChar – печать символа на экран. Принимает в качестве аргумента символ, а возвращает IO ().

  ```
  putChar :: Char -> IO ()
  ```

- return – возвращает значения без ожидания интерактивных действий

  ```
  return :: a -> IO a
  ```

Последовательность операций ввода/вывода

Последовательность действий ввода/вывода соединяются зарезервированным словом do.

```
fun :: IO (Char,Char)
fun  = do x -> getChar
          getChar
          y -> getChar
          return (x,y)
```

Вывод на экран

Вывод текстовой строки на экран:

```
putStr        :: String -> IO ()
putStr []      = return ()
putStr (x:xs) = do putChar x
                   putStr xs
```

Вывод текстовой строки на экран с новой строки:

```
putStrLn    :: String -> IO ()
putStrLn xs = do putStr xs
                 putStr '\n'
```

Вывод на экран

Для компиляции программы необходимо определить функцию main, например в файле filename. Данная функция будет выполнена после создания командного файла.

```
main :: IO String
main = do foo <- putStrLn "Hello, what's your name?"
          name <- getLine
          putStrLn ("Hey " ++ name )
```

Компилятор ghc. В командной строке Linux набираем:

```
localhost$ ghc --make filename
localhost$ ./filename
Archi
Hey Arch!
```

Пример программмы

Программа считывает с клавиатуры строки до тех пор пока и открывает в зашифрованной строке выбранные символы.

```
main = hangman
hangman :: IO ()
hangman  =
   do putStrLn "Think of a word: "
      word <- sgetLine
      putStrLn "Try to guess it:"
      guess word

sgetLine :: IO String
sgetLine  = do x <- getCh
               if x == '\n' then
                  do putChar x
                     return []
                else
                  do putChar '-'
                     xs <- sgetLine
                     return (x:xs)
```

Пример программмы

```
guess :: String -> IO ()
guess word =
   do putStr "> "
      xs <- getLine
      if xs == word then
         putStrLn "You got it!"
       else
         do putStrLn (diff word xs)
            guess word
```

Функция diff

```
diff      :: String -> String -> String
diff xs ys =
   [if elem x ys then x else '-' | x <- xs]
```

Контрольные задания

- Программа считывает введенную строку символов и заменяет все гласные буквы на символ *.
- Программа считывает последовательно 2 строки и выводит объединение двух строк.
- Программа считывает последовательность символов до тех пор, пока не будет нажата последовательность qwerty.
- Считать с клавиатуры 2 числа и найти их сумму. Результат вывести на экран.