Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

КУРСОВАЯ РАБОТА
по дисциплине «Языки и средства функционального программирования»

Калькулятор

Выполнил:
Студент гр. 3530904/80003 Никурашин А.С.

Принял: Лукашин А. А.

Санкт-Петербург 2019 г.

Оглавление

Введение.....3
Описание задачи.....4
Текст программы.....5
Скриншоты программы.....6
Вывод.....7
Ссылка на репозиторий.....8

Введение

Функциональное программирование – это ветвь программирования, при котором программирование ведется с помощью определения функций.

В функциональном программировании нет ни процедур, ни циклов, нет даже переменных. Почти одни только функции. Функциональное программирование обладает рядом очень существенных преимуществ, которые не только позволяют ему существовать наряду с традиционным программированием, но и иметь своих поклонников, свою нишу задач и хорошие перспективы на будущее.

Ветвь программирования, начатая созданием Лиспа, понемногу развивалась с начала 60-х годов 20 века и привела к появлению целой плеяды очень своеобразных языков программирования, которые удовлетворяли всем требованиям, необходимым для исполнения программ несколькими параллельными процессорами. Во-первых, алгоритмы, записанные с помощью этих языков, допускают сравнительно простой анализ и формальные преобразования программ, а во-вторых, отдельные части программ могут исполняться независимо друг от друга. Языки, обладающие такими замечательными свойствами – это и есть языки функционального программирования. Помимо своей хорошей приспособленности к параллельным вычислениям языки функционального программирования обладают еще рядом приятных особенностей. Программы на этих языках записываются коротко, часто много короче, чем в любом другом традиционном (императивном) языке. Описание алгоритмов в функциональном стиле сосредоточено не на том, как достичь нужного результата (в какой последовательности выполнять шаги алгоритма), а больше на том, что должен представлять собой этот результат.

Единственный недостаток функционального стиля программирования состоит в том, что этот стиль не универсальный. Многие действительно последовательные процессы, такие как поведение программных моделей в реальном времени, игровые и другие программы, организующие взаимодействие компьютера с человеком, не выразимы в функциональном стиле.

Функциональное программирование позволяет несколько по-иному взглянуть вообще на процесс программирования, а некоторые приемы программирования, которые, предназначены для написания программ в чисто функциональном стиле, могут с успехом использоваться и в традиционном программировании.

Описание задачи

Реализовать калькулятор арифметических выражений на функциональном языке Haskell.

Текст программы

```
import Text.Parsec
import Text.Parsec.String
import Text.Parsec.Token
import Text.Parsec.Language
import Text.Parsec.Expr

lexer :: TokenParser()
lexer = makeTokenParser (javaStyle { opStart  = oneOf "+-*/"
                                   , opLetter = oneOf "+-*/" })

parseNumber :: Parser Double
parseNumber = do
  num <- naturalOrFloat lexer
  case num of
    Left int -> return $ fromIntegral int
    Right n -> return $ n

parseExpression :: Parser Double
parseExpression = (flip buildExpressionParser) parseItem $  [
    [ Prefix (reservedOp lexer "-" >> return negate) ],
    [ Infix  (reservedOp lexer "+" >> return (+)) AssocLeft,
      Infix  (reservedOp lexer "-" >> return (-)) AssocLeft],
    [ Infix  (reservedOp lexer "*" >> return (*)) AssocLeft,
      Infix  (reservedOp lexer "/" >> return (/)) AssocLeft]
  ]

parseItem :: Parser Double
parseItem = parens lexer parseExpression <|> parseNumber

parseInput :: Parser Double
parseInput = do
  n <- parseExpression
  eof
  return n

calculate :: String -> String
calculate s =
  case ret of
    Left e -> "error: " ++ (show e)
    Right n -> "answer: " ++ (show n)
  where
    ret = parse parseInput "" s

eachLine :: (String -> String) -> (String -> String)
eachLine calculate = unlines . (map calculate) . lines

main :: IO ()
main = interact (eachLine calculate)
```

Скриншоты программы

Вывод

В ходе выполнения курсовой работы были улучшены навыки разработки программ на языке haskell

В данном курсовом проекте был реализован калькулятор, который работает на стандартных потоках вводах и вывода, калькулятор поддерживает операции сложения, вычитания, деления и умножения. Использование принципов функционального программирования является очень удобным и продуктивным при написании программ.

Ссылка на репозиторий

https://github.com/mycelium/hsse-fp-2019-2/tree/3530904/80005_Nikurashin-Alexey/tasks