Odwrócona notacja polska

Przy użyciu języka Haskell

Jakub.J i Kamil.K

Politechnika Krakowska

February 13, 2020

Struktura prezentacji

1. Algebra elementarna
   - "Szkolna" algebra
   - Notacja polska
   - Odwrotna notacja polska
2. Odwrotna notacja polska - implementacja
   - Kod w Haskellu

"Szkolna" algebra

Podczas nauki algebry elementarnej w szkole zwykliśmy zapisywać wyrażenie algebraiczne w notacji infiksowej, np. $8 - (2 + 3) * 2$. Dodawanie $(+)$, odejmowanie $(-)$ oraz mnożenie $(*)$ to operatory infiksowe. Przy przyjętej matematycznej konwencji kolejności wykonywania działań najpierw wykonujemy wyrażenie w nawiasie, działanie mnożenia a na końcu działanie odejmowania. Korzyścią z takiego zapisu jest łatwe wyobrażenie sobie tych działań w naszych umysłach, ale istnieje dość duża wada, którą niesie za sobą używanie notacji infiksowej.

Notacja polska

Notacja Polska nie jest tak oczwista jak notacja infiksowa. Często określa się ją jako notację "przedrostkową", bo operatory umieszcza się w niej przed argumentami, np: + 2 2. Daje w wyniku 4. Jest ona jednak niewspółmiernie bardziej czytelna dla jednostek arytmetyczno logicznych. Notacja polska jest bliska naturalnemu sposobowi wyrażania działań, w którym zazwyczaj najpierw podaje się czynność, a następnie dopełnia wyrażenie wskazaniem rzeczy, do których czynność się odnosi. Stąd zapis przedrostkowy stał się podstawą edukacyjnego języka programowania Logo a także języków Tcl i Lisp.

Odwrotna notacja polska

W porównaniu do zapisu wyrażeń w notacji polskiej, istnieje także pochodna podobnego zapisu zwana Odwrotną notacją polską. Znak wykonywanej operacji umieszczony jest po działaniach (operandach) a nie przed operandami jak w zwykłej notacji polskiej. Korzyścią z zastosowania odwrotnej notacji polskiej jest całkowita rezygnacja z nawiasów, gdyż można w sposób unikatowy określić kolejność wykonywania działań.

Kod w Haskellu

18 lines (13 sloc) | 619 Bytes — Raw | Blame | History

```haskell
main :: IO ()
main = print(solveRPN "2 3.5 +")


solveRPN :: String -> Double

solveRPN = head . foldl foldingFunction [] . words
    where foldingFunction (x:y:ys) "*" = (y * x):ys
          foldingFunction (x:y:ys) "+" = (y + x):ys
          foldingFunction (x:y:ys) "-" = (y - x):ys
          foldingFunction (x:y:ys) "/" = (y / x):ys
          foldingFunction (x:y:ys) "^" = (y ** x):ys
          foldingFunction (x:xs) "ln" = log x:xs
          foldingFunction xs "sum" = [sum xs]
          foldingFunction xs numberString = read numberString:xs
```

Czym jest „Odnajdywanie wzorca"

Odnajdywanie wzorca, czy też bardziej popularna nazwa – „pattern matching" polega na sprecyzowaniu wzorców, dla których niektóre dane powinny być zgodne, sprawdzeniu wzorców oraz rozkładzie danych zgodnie z wzorcami. Możliwe jest definiowanie oddzielnych funkcji dla różnych wzorców. Umożliwia to przekierowywanie z poszczególnych typów danych do innych, np. z int'a do list, znaków itp.

Czym jest „Odnajdywanie wzorca"

Oczywiście funkcja tego rodzaju mogłaby w bardzo prosty sposób być zastąpiona funkcja IF, jednakże użycie wzorców jest idealnym rozwiązaniem, gdy mamy więcej funkcji.
Oto przykładowy wzorzec przypisujący nazwę dnia tygodnia do numeru dnia tygodnia:

```
week:: ( Integral a) => a -> String
        week 1 = " Poniedzialek "
        week 2 = " Wtorek "
        week 3 = " Sroda "
        week 4 = " Czwartek "
        week 5 = " Piatek "
        week 5 = " Sobota "
        week 5 = " Niedziela "
        week x = " To nie jest liczba z przedzialu 1-7"
```

Kod w Haskellu

main.hs

c: > Users > HUAWEI > Desktop > SemestrV > ProgramowanieFunkcyjne > ReversePolishNotationProject > main.hs

```haskell
main :: IO ()
main = print(solveRPN "10 2 ^")


solveRPN :: String -> Double

solveRPN = head . foldl foldingFunction [] . words
    where foldingFunction (x:y:ys) "*" = (y * x):ys
          foldingFunction (x:y:ys) "+" = (y + x):ys
          foldingFunction (x:y:ys) "-" = (y - x):ys
          foldingFunction (x:y:ys) "/" = (y / x):ys
          foldingFunction (x:y:ys) "^" = (y ** x):ys
          foldingFunction (x:xs) "ln" = log x:xs
          foldingFunction xs "sum" = [sum xs]
          foldingFunction xs numberString = read numberString:xs

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Haskell Run

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\HUAWEI> stack runhaskell "c:\Users\HUAWEI\Desktop\SemestrV\ProgramowanieFunkcyjne\ReversePolishNotationProject\main.hs"
100.0
PS C:\Users\HUAWEI>
```