Министерство науки и высшего образования Российской Федерации
Национальный научно-исследовательский университет ИТМО
Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по дисциплине

«Функциональное программирование».

Вариант №Prefix Tree / Set.

Работу выполнил:
Афанасьев Кирилл Александрович,
Студент группы P3306.
Преподаватели:
Новоселов Борис Сергеевич,
Пенской Александр Сергеевич.

Санкт-Петербург, 2024

Оглавление

Задание

Цель: освоиться с построением пользовательских типов данных, полиморфизмом, рекурсивными алгоритмами и средствами тестирования (unit testing, property-based testing).

В рамках лабораторной работы вам предлагается реализовать одну из предложенных классических структур данных (список, дерево, бинарное дерево, hashmap, граф...).

Требования:

1. Функции:
    - добавление и удаление элементов;
    - фильтрация;
    - отображение (map);
    - свертки (левая и правая);
    - структура должна быть моноидом.
2. Структуры данных должны быть неизменяемыми.
3. Библиотека должна быть протестирована в рамках unit testing.
4. Библиотека должна быть протестирована в рамках property-based тестирования (как минимум 3 свойства, включая свойства моноида).
5. Структура должна быть полиморфной.
6. Требуется использовать идиоматичный для технологии стиль программирования. Примечание: некоторые языки позволяют получить большую часть API через реализацию небольшого интерфейса. Так как лабораторная работа про ФП, а не про экосистему языка -- необходимо реализовать их вручную и по возможности -- обеспечить совместимость.

Исходный код программы

GitHub: https://github.com/Zerumi-ITMO-Related/fp2_041024_pre-set

Ключевые элементы реализации:

```haskell
-- Definition of Trie data structure
data Trie a = Node [(a, Trie a)] Bool deriving (Eq, Show, Generic)
```

Реализованный набор функций:

```haskell
module TrieModule
  ( Trie(..),
    empty,
    insert,
    remove,
    member,
    filter,
    _foldl,
    _foldr,
    _map,
    toList,
    fromList
  )
```

Unit и Property Based тестирование:

```haskell
-- Unit & Property-based tests for `Trie` operations
spec :: Spec
spec = do
 describe "TrieModule" $ do
  -- Validity property tests
  it "ensures that empty Trie is valid" $
    shouldBeValid (empty :: Trie Char)

  it "ensures that insert maintains validity" $
    forAllValid $ \(word :: String) ->
      forAllValid $ \trie ->
        shouldBeValid (TrieModule.insert word trie)

  it "ensures that remove maintains validity" $
    forAllValid $ \(word :: String) ->
      forAllValid $ \trie ->
        shouldBeValid (remove word trie)

  it "ensures that filter maintains validity" $
    forAllValid $ \(trie :: Trie Char) ->
```

```
      shouldBeValid (TrieModule.filter (\w -> length w > 2) trie)

  it "ensures that toList maintains validity" $
    forAllValid $ \(trie :: Trie Char) ->
      shouldBeValid (toList trie)

  it "ensures that fromList maintains validity" $
    forAllValid $ \words ->
      shouldBeValid (fromList (words :: [String]))

  it "ensures that member maintains validity" $
    forAllValid $ \(word :: String) ->
      forAllValid $ \trie ->
        shouldBeValid (member word trie)

  it "ensures that trie has a neutral element that doesn't affect the trie when inserted" $
    forAllValid $ \trie ->
      insert "" trie `shouldBe` trie

  it "unit testing numbers trie, polymorphic test" $ do
    let numTrie = empty :: Trie Int
    let updated = insert [1] $ insert [1, 2, 3] numTrie
    toList updated `shouldBe` [[1], [1, 2, 3]]
    member [1, 2, 3] updated `shouldBe` True
    member [1, 2, 4] updated `shouldBe` False
    let removed = remove [1, 2, 3] updated
    member [1, 2, 3] removed `shouldBe` False
```

Настроенный GitHub Actions CI:

```
name: Haskell Updated CI

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

permissions:
  contents: read

jobs:
  hlint:
```

```yaml
  name: Run lint
  runs-on: ubuntu-latest
  steps:
  - uses: actions/checkout@v4

  - name: 'Set up HLint'
    uses: haskell-actions/hlint-setup@v2

  - name: 'Run HLint'
    uses: haskell-actions/hlint-run@v2
    with:
      path: src/
      fail-on: warning
build-test:
  name: Build & Test
  runs-on: ubuntu-latest
  steps:
    - uses: actions/checkout@v4
    - uses: haskell-actions/setup@v2
      with:
        ghc-version: '9.6.6'
        enable-stack: true
        stack-version: 'latest'
    - run: stack --no-terminal test --fast
```

Вывод

Во время выполнения данной лабораторной работы я ознакомился с построением собственных типов данных в Haskell, а также ознакомился с очень удобным набором для Validity-based тестирования, позволяющим автоматически генерировать тестовую нагрузку для реализованного типа данных.