КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра прикладних інформаційних систем

Звіт до лабораторної роботи №2

з курсу

«Функціональне програмування»

Студентки 4 курсу
групи ПП-41
спеціальності 122 «Комп'ютерні науки»
ОП «Прикладне програмування»
Штонь Софії Максимівни

Викладач:
Пирог М. В.

Київ – 2023

Тема роботи: Визначення нових типів даних.

Мета роботи: Ознайомитися з механізмом визначення нових типів даних в Haskell. Розширити навички використання існуючих простих типів даних. Поглибити знання механізмів обчислення з невизначеним результатом.

Теоретичні відомості

Тема №2 «Основи мови Haskell» має на меті розкриття основних аспектів програмування мовою Haskell. Для цього описуються особливості системи програмування Haskell Platform, процес визначення функцій за допомогою рівнянь, пояснюються типи функцій, основи роботи зі списками та типами даних. На самостійне опрацювання виноситься питання визначення нових типів даних для закріплення та розширення набутих знань з основ Haskell.

Для поглиблення знань в області базового програмування мовою Haskell, а саме визначення нових типів даних, слід особливу увагу звернути на типи і класи, як такі, до яких належать класичні типи даних. Визначення нових типів даних має на увазі і визначення класу, до якого вони будуть належати. Крім того, необхідно опрацювати механізми обчислення з невизначеним результатом, які прямо відносяться до визначення типів даних. Слід дослідити процес виводу класів типів та можливість рекурсивного визначення типів даних (рекурсивні структури даних). Доцільно засвоїти механізм використання синонімів типів та конструкторів типів, а також призначення та особливості кожної технології.

Завдання для виконання

Перевизначити існуючий клас простих типів даних, визначений за замовчуванням та дати їм нові імена, використовуючи механізм синонімів типів із застосуванням параметрів типів.

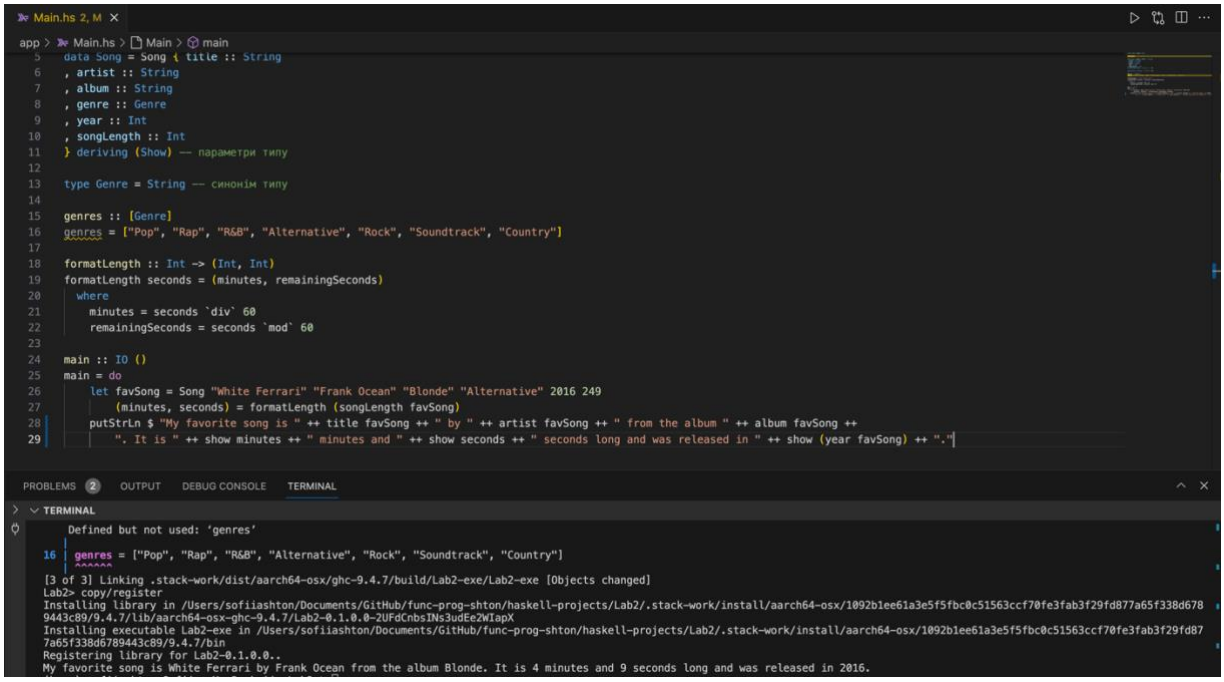
Хід роботи

Код програми (Haskell):

```
module Main (main) where

import Lib
```

```haskell
data Song = Song { title :: String
, artist :: String
, album :: String
, genre :: Genre
, year :: Int
, songLength :: Int
} deriving (Show) -- параметри типу

type Genre = String -- синонім типу

genres :: [Genre]
genres = ["Pop", "Rap", "R&B", "Alternative", "Rock", "Soundtrack", "Country"]

formatLength :: Int -> (Int, Int)
formatLength seconds = (minutes, remainingSeconds)
  where
    minutes = seconds `div` 60
    remainingSeconds = seconds `mod` 60

main :: IO ()
main = do
    let favSong = Song "White Ferrari" "Frank Ocean" "Blonde" "Alternative" 2016 249
        (minutes, seconds) = formatLength (songLength favSong)
    putStrLn $ "My favorite song is " ++ title favSong ++ " by " ++ artist favSong ++ " from the album " ++ album favSong ++
        ". It is " ++ show minutes ++ " minutes and " ++ show seconds ++ " seconds long and was released in " ++ show (year
favSong) ++ "."
```

Результат роботи програми:



Висновок: Отже, у ході цієї лабораторної роботи було проведено ознайомлення з визначенням нових типів даних в мові програмування Haskell. Було написано програму, яка перевизначає існуючий клас простих типів даних, використовуючи механізми синонімів типів та параметрів типів.

Контрольні запитання

1. *Яким чином оголошується новий, не існуючий за замовчуванням тип даних мовою Haskell?*

Новий тип даних оголошується за допомогою ключового слова data. Наприклад:

data Shape = Circle Float Float Float | Rectangle Float Float Float Float

або

data Color = Red | Green | Blue

2. *Що таке конструктори значень та яку роль вони відіграють у визначенні типів даних?*

Це іменовані шаблони, що використовуються для створення значень певного типу даних. Кожен конструктор визначає різний спосіб створення об'єктів цього типу. Наприклад:

data Point = Point Double Double,

де Point - тип даних та конструктор значень, який приймає два аргументи типу Double і створює значення типу Point. Його можна використовувати так:

```
myPoint :: Point
myPoint = Point 2.0 3.0
```

Конструктори значень грають важливу роль у визначенні типів даних, оскільки вони визначають, як можна створювати значення цього типу. Вони також дозволяють задавати поля та властивості для складніших типів даних.

3. Яким чином використовуються конструктори типів?

Конструктори типів використовуються для створення нових типів даних або визначення нових конструкторів для існуючих типів. Вони вказують структуру та параметри, які може приймати значення цього типу.

4. Яку роль у визначенні нових типів відіграють класи та яким чином відбувається визначення нових класів типів?

Класи у Haskell визначають інтерфейси, а типи реалізують ці інтерфейси. Класи типів визначають набір функцій чи методів, які повинні бути реалізовані для конкретного типу. Наприклад, клас типів Show дозволяє представляти значення у вигляді рядка:

```
class Show a where
  show :: a -> String
```

Це визначення говорить, що для типу a можна створити екземпляр класу Show, який реалізує функцію show, яка приймає значення типу a і повертає його у вигляді рядка. Екземпляр для типу Person:

```
data Person = Person { name :: String, age :: Int }
instance Show Person where
  show (Person name age) = "Person {name: " ++ name ++ ", age: " ++ show age ++ "}"
```

5. *В чому суть механізму обчислення з невизначеним результатом та в яких випадках його доцільно використовувати?*

Механізм обчислення з невизначеним результатом часто використовується за допомогою типу Maybe. Наприклад, функція, яка шукає елемент у списку:

```
findElement :: Eq a => a -> [a] -> Maybe a
findElement _ [] = Nothing
findElement x (y:ys)
  | x == y    = Just x
  | otherwise = findElement x ys
```

Ця функція повертає Just x, якщо елемент x знайдено у списку, і Nothing, якщо його немає. Використовуючи Maybe, можна явно вказати можливість відсутності значення, що полегшує обробку таких ситуацій. Такий підхід робить код більш безпечним, оскільки він вимагає явно обробляти випадок відсутності значення, замість того, щоб просто ввести null або ігнорувати помилку.