КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра прикладних інформаційних систем

Звіт до лабораторної роботи №5

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: Налагодження програм на Haskell.

Мета роботи: Ознайомитися з процесом налагодження програм на Haskell. Розширити навички використання функторів та монад при програмуванні мовою Haskell. Поглибити знання технологій компіляції програм на Haskell.

Теоретичні відомості

Тема №5 «Ліниві обчислення. Функтори та монади» має на меті розкриття основних аспектів складу стандартних класів та використання нескінченних структур даних для програмування мовою Haskell. Для цього описуються особливості енергійної та лінивої схеми обчислень, розглядаються нескінченні структури даних, функтори, монади та послідовні обчислення, а також процес введення, виведення і компіляції програм на Haskell. На самостійне опрацювання виноситься питання налагодження програмного коду на Haskell.

Для поглиблення знань в області функціонального програмування на Haskell, а саме в налагодженні програм, слід особливу увагу звернути на різновиди методик налагодження. Необхідно розглянути інструментальні засоби налагодження програм, написаних функціональними мовами взагалі та мовою Haskell зокрема. Слід приділити увагу текстовому представленню даних, що спрощує процедуру налагодження програм, а також на процес віддаленого налагодження.

Завдання для виконання

Використовуючи мову Haskell, реалізувати програма «вгадування» задуманого числа (у вигляді: «Задумайтесь число від 0 до 99 Задумане менше, ніж 50? (Yes, по)». Якщо ні, то «Задумане менше, ніж 75? (Yes, по)» і т.д.). Обрати найбільш вдалий інструмент та методику налагодження. Провести налагодження реалізованої програми.

Хід роботи

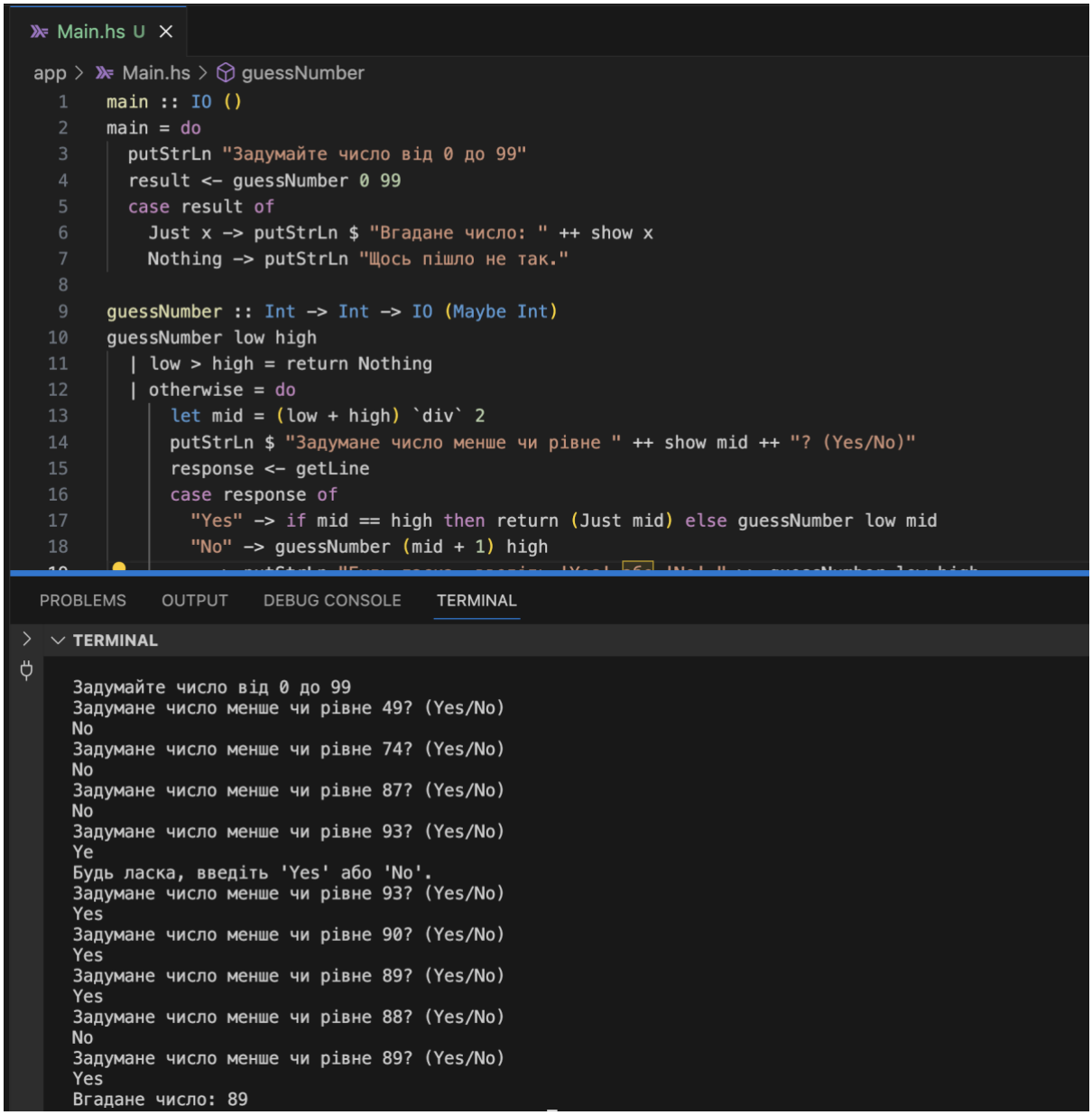
Код програми (Haskell):

```
main :: IO ()
main = do
  putStrLn "Задумайте число від 0 до 99"
  result <- guessNumber 0 99
```

```
  case result of
    Just x -> putStrLn $ "Вгадане число: " ++ show x
    Nothing -> putStrLn "Щось пішло не так."

guessNumber :: Int -> Int -> IO (Maybe Int)
guessNumber low high
  | low > high = return Nothing
  | otherwise = do
      let mid = (low + high) `div` 2
      putStrLn $ "Задумане число менше чи рівне " ++ show mid ++ "? (Yes/No)"
      response <- getLine
      case response of
        "Yes" -> if mid == high then return (Just mid) else guessNumber low mid
        "No" -> guessNumber (mid + 1) high
        _ -> putStrLn "Будь ласка, введіть 'Yes' або 'No'." >> guessNumber low high
```

Результат роботи програми:



Висновок: Отже, у ході цієї лабораторної роботи було проведено ознайомлення з використанням функторів та монад при програмуванні мовою Haskell.

Контрольні запитання

1. В чому полягає особливість налагодження функціональних програм?

Особливості налагодження функціональних програм полягають у тому, що функціональні мови, такі як Haskell, використовують незмінність даних. Це означає, що дані, коли вони одного разу створені, залишаються незмінними. Такий підхід полегшує розуміння коду та виключає багато проблем, пов'язаних з небезпечними змінами стану. Також, в функціональних програмах відсутні ефекти побічних дій. Це означає, що функції не мають побічних ефектів, таких як зміна глобальних змінних чи

стану програми. Відсутність побічних ефектів робить налагодження більш передбачуваним та полегшує зрозуміння функціонального коду.

2. Які інструментальні засоби забезпечують процес налагодження програмного коду на Haskell?

Для простішого відлагодження та трасування можна використовувати модуль Debug.Trace. Функція trace допомагає виводити повідомлення під час виконання програми. Ще однією альтернативою є бібліотека safe, яка надає безпечніші версії деяких функцій Prelude, що сприяє більш гнучкій обробці помилок.

3. Які характерні особливості механізму декларативного налагодження програм на Haskell?

Однією з ключових переваг є безпека типів, яка дозволяє виявляти помилки на етапі компіляції та надає високий рівень впевненості у вірності програми. Підказки про типи та можливі помилки часто допомагають ідентифікувати та виправляти проблеми. Чистота функцій є ще однією важливою рисою. Функції в Haskell є чистими, що дозволяє уникнути побічних ефектів та полегшує відлагодження через передбачувану поведінку.

4. Як можливо використати традиційну методику налагодження при використанні лінивих обчислень?

Використання традиційної методики налагодження в поєднанні з лінивими обчисленнями у Haskell може вимагати специфічних підходів для оптимального відлагодження. Для виведення проміжних результатів можна використовувати Trace-функції, такі як Debug.Trace, або вставляти виведення безпосередньо в код за допомогою putStrLn. GHCi може слугувати корисним інструментом для інтерактивної взаємодії та експериментів з кодом, дозволяючи тестувати вирази та відстежувати виконання програми.

5. Які існують методики налагодження програм функціональних мов?

Однією з загальних методик є виведення в консоль (Debugging by Printing), що передбачає використання функцій виведення для виведення проміжних результатів обчислень. Також ефективним є застосування трасування (Tracing), використовуючи

засоби, які дозволяють вставляти сліди в код та виводити інформацію про хід виконання програми. Інтерактивне тестування (Interactive Testing) через інтерактивні середовища, такі як GHCi в Haskell, дозволяє тестувати функції та вирази на льоту, спрощуючи відлагодження. Explicit Condition Checking може бути використана для виявлення та виведення повідомлень про помилки під час виконання.