КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра прикладних інформаційних систем

Звіт до лабораторної роботи №8

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: Подання рекурсивних функцій при редукції на графах.

Мета роботи: Ознайомитися з правилами подання рекурсивних функцій при редукції на графах. Розширити навички графічного представлення функціональних програм. Поглибити знання в області абстрагування та виявити різницю абстрагування в імперативному та функціональному програмуванні.

Теоретичні відомості

Тема №8 «Абстрагування у функціональному програмуванні. Довговічність коду» має на меті розкриття основних аспектів абстракції у функціональному програмуванні. Для цього описуються особливості використання імперативних програм, як функцій, розглядається абстрагування від змінних, оптимізація Каррі та збереження аплікативних під виразів при абстрагуванні. На самостійне опрацювання виноситься питання подання рекурсивних функцій при редукції на графах у зв'язку з необхідність засвоєння різниці абстракції в імперативному і функціональному підході програмування, зважаючи на те, що застосування методу процедурної абстракції передбачає використання рекурсивних функцій в імперативному підході.

Для поглиблення знань в області абстрагування та в поданні рекурсивних функцій слід особливу увагу звернути на правила перетворення графів для основних комбінаторів. Крім того, слід розглянути послідовність перетворення рекурсивного блоку в аплікативний вираз, особливості перетворення графа для Y-комбінатора та спосіб уникнення застосування Y-комбінатора, що має на увазі утворення циклічних посилань в графі.

Завдання для виконання

Функція gcd обчислення найбільшого спільного дільника двох натуральних чисел може бути представлена рівняннями gcd m 0 = m gcd m n = gcd n (m 'mod' n). Зробіть графічне представлення виразу gcd 9 10 і проведіть редукції в отриманому графі до приведення його до нормальної форми.

Хід роботи

Код програми (Haskell):

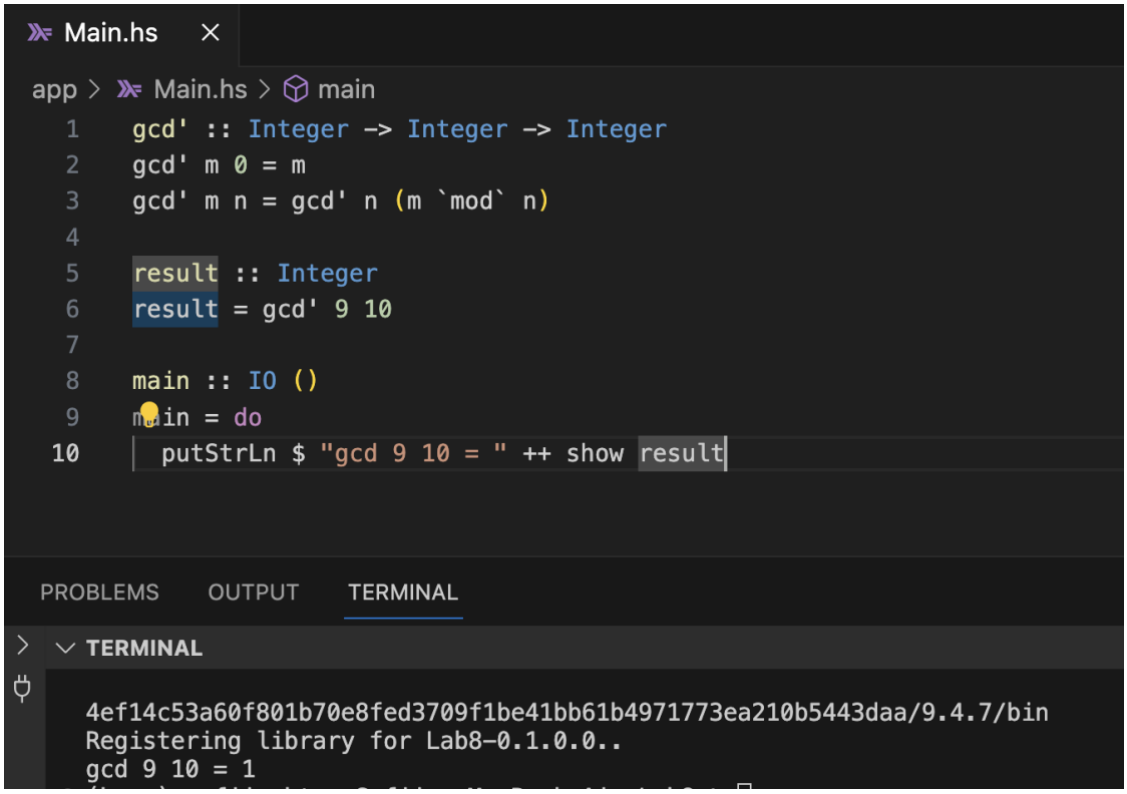
```
gcd' :: Integer -> Integer -> Integer
gcd' m 0 = m
```

```haskell
gcd' m n = gcd' n (m `mod` n)

result :: Integer
result = gcd' 9 10

main :: IO ()
main = do
  putStrLn $ "gcd 9 10 = " ++ show result
```

Результат роботи програми:



Висновок: Отже, у ході цієї лабораторної роботи було проведено ознайомлення з правилами подання рекурсивних функцій при редукції на графах. Було розширено навички графічного представлення функціональних програм та поглиблено знання в області абстрагування.

Контрольні запитання

1. *Як підвищується ефективність роботи по перетворенню графів?*

Ефективність роботи по перетворенню графів може бути підвищена за допомогою таких оптимізаційних технік, як виявлення та усунення зайвих обчислень, розпаралелювання обчислень або використання лінивих обчислень.

2. Як можливо позбутися від рекурсії при, якщо вираз містить рекурсивні блоки?

Для позбавлення виразу від рекурсивних блоків можна використовувати техніки які, наприклад, використовують ітеративні підходи замість рекурсії, тим самим уникнувши глибоких стеків викликів.

3. Як виглядає правило, що визначає Y-комбінатор?

Y-комбінатор використовує самопосилання для створення рекурсивних функцій у лямбда-виразі:

```
y :: (a -> a) -> a

y = \f -> (\x -> f (x x)) (\x -> f (x x))
```

4. Яким чином операції, необхідні, щоб позбутися рекурсії можливо представити у програмному вигляді?

Операції, необхідні для усунення рекурсії, можна представити у програмному вигляді за допомогою циклів або інших технік, які замінюють рекурсивні виклики ітеративними або хвостовими викликами.

5. Яким чином можна зобразити графічно правило перетворення графа для Y-комбінатора?

Графічно правило перетворення графа для Y-комбінатора можна зобразити як лямбда-вираз, де вузли представляють функції, а ребра відображають залежності між ними, включаючи самопосилання, що характерно для Y-комбінатора.