

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені
ТАРАСА ШЕВЧЕНКА**



ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра прикладних інформаційних систем

Звіт до лабораторної роботи №6

з курсу

«Функціональне програмування»

*Студентки 4 курсу
групи ПП-41*

спеціальності 122 «Комп'ютерні науки»

ОП «Прикладне програмування»

Штось Софії Максимівни

*Викладач:
Пирог М. В.*

Київ – 2023

Тема роботи: Подання програм при розширеному лямбда-численні.

Мета роботи: Ознайомитися з поданням програм при розширеному лямбда-численні. Розширити навички застосування чистого лямбда-числення. Поглибити знання технологій представлення конструкцій мовою програмування Haskell.

Теоретичні відомості

Тема №6 «Лямбда-числення в аналізі та синтезі інформаційних систем» має на меті розкриття основних аспектів застосування математичного підґрунтя функціональної парадигми програмування. Для цього описуються особливості подання виразів у лямбда-численні, рекурсії в ньому та розглядається чисте лямбда-числення, а також наводяться відомості про нормальну та слабку заголовну нормальну форму. На самостійне опрацювання виносяться питання розширеного лямбда-числення та особливостей подання програм у ньому, як такому, що базується на принципах чистого лямбда-числення та дозволяє закріпити і поглибити знання цих принципів.

Для поглиблення знань в області лямбда-числення, а саме в області розширеного лямбда-числення слід особливу увагу звернути на загальні принципи та основи розширеного лямбда-числення, застосування правил редукції в умовах існування рекурсивних визначень. Також необхідно звернутися до питань класифікації конструкцій в розширеному лямбда-численні (особливо виділяють константи) і можливості зручного представлення конструкцій розширеного лямбда-числення у функціональних програмах.

Завдання для виконання

Необхідно реалізувати функцію, що за лямбда-виразом рекурсивної функції будує нерекурсивну, яка є еквівалентною їй за допомогою Y-комбінатору, що є вбудованою функцією.

Хід роботи

Код програми (Haskell):

```
module Main (main) where

import Lib
```

```

main :: IO ()
main = print (fac 6)

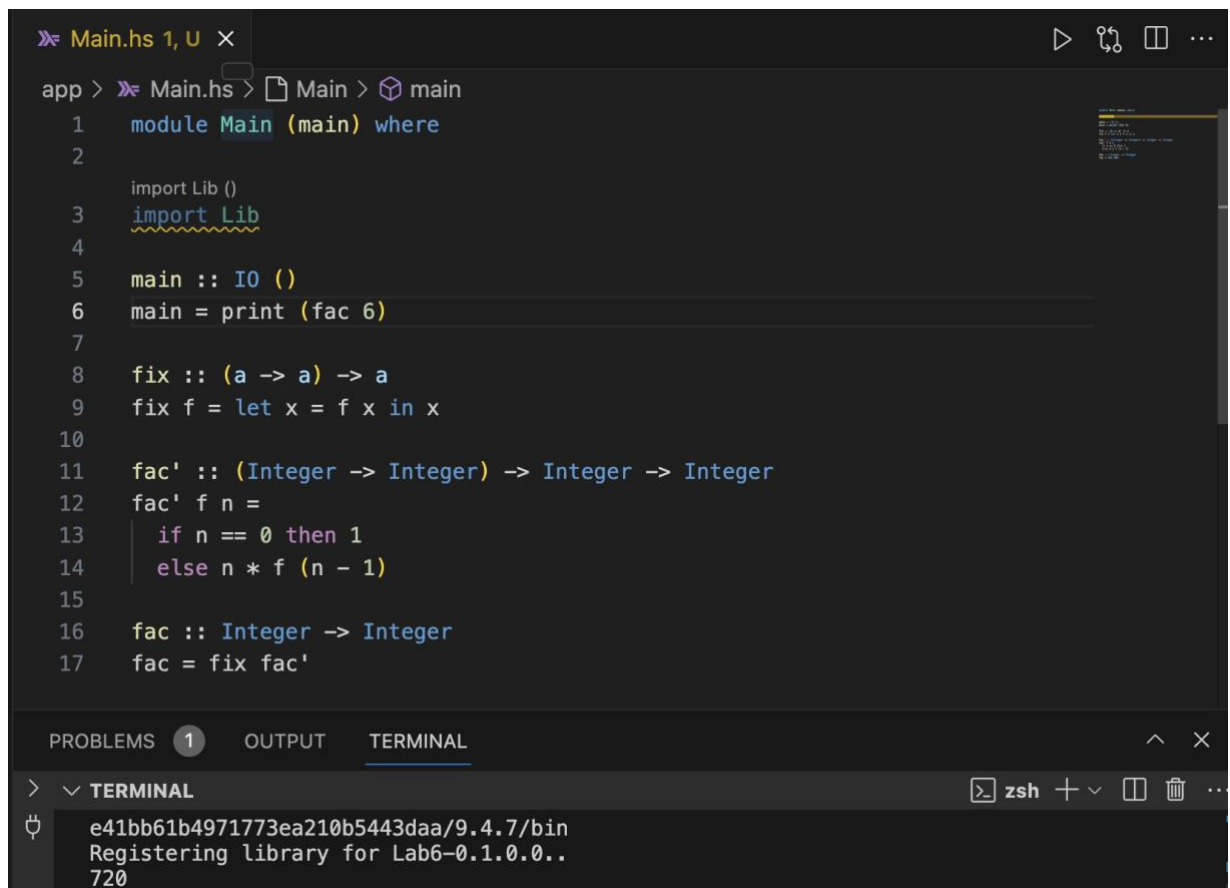
fix :: (a -> a) -> a
fix f = let x = f x in x

fac' :: (Integer -> Integer) -> Integer -> Integer
fac' f n =
  if n == 0 then 1
  else n * f (n - 1)

fac :: Integer -> Integer
fac = fix fac'

```

Результат роботи програми:



The screenshot shows a Haskell IDE with a file named `Main.hs`. The code in the editor matches the code block above. The bottom panel shows the `TERMINAL` tab with the following output:

```

e41bb61b4971773ea210b5443daa/9.4.7/bin
Registering library for Lab6-0.1.0.0..
720

```

Висновок: Отже, у ході цієї лабораторної роботи було проведено ознайомлення з поданням програм при розширеному лямбда-численні. Було розширено навички

застосування чистого лямбда-числення. Також, було поглиблено знання технологій представлення конструкцій мовою програмування Haskell.

Контрольні запитання

1. Які переваги має розширене лямбда-числення в порівнянні із чистим в контексті вирішення практичних задач?

По-перше, розширене лямбда-числення може надавати зручний механізм для рекурсії, що дозволяє визначати рекурсивні функції без додаткових ускладнень. Додавання констант до мови дозволяє легше працювати з фіксованими значеннями, що важливо в практичному програмуванні. Крім того, розширене лямбда-числення може включати розширення для визначення та використання додаткових типів даних. Це розширення полегшує роботу з різноманітними структурами даних, що є необхідним в реальних програмах.

2. Яким чином можливо більш точно дослідити семантику виконання функціональних програм, використовуючи розширене лямбда-числення?

Вивчення семантики виконання функціональних програм за допомогою розширеного лямбда-числення дозволяє більш точно досліджувати роботу програм, оскільки це розширення надає більше можливостей для визначення та аналізу властивостей функцій. Зокрема, додавання констант, рекурсивних визначень та додаткових типів даних розширює експресивність мови, дозволяючи точніше моделювати і виражати концепції в програмах.

3. Які види конструкцій існує в розширеному лямбда-численні?

В розширеному лямбда-численні визначаються різноманітні конструкції, такі як рекурсивні визначення, константи та розширення для роботи з додатковими типами даних. Ці конструкції розширюють можливості мови порівняно з чистим лямбда-численням, надаючи програмістам більше інструментів для виразного програмування.

4. В чому полягає особливість вираження конструкцій функціональних мов програмування шляхом застосування розширеного лямбда-числення?

Особливість вираження конструкцій функціональних мов програмування за допомогою розширеного лямбда-числення полягає в можливості абстрактного виразу складних концепцій. Додавання рекурсивних визначень, констант та розширень для роботи з типами даних робить код більш зрозумілим, гнучким та виразним.

5. *Яким чином проводиться опис конструкцій синтаксичного дерева, використовуючи новий тип даних мовою Haskell?*

Опис конструкцій синтаксичного дерева може бути проведений за допомогою нових типів даних у мові Haskell. Використання алгебраїчних типів даних дозволяє точно моделювати структури конструкцій та їх взаємодію. Такий підхід полегшує розуміння та роботу з синтаксичним деревом програми, забезпечуючи чіткість та розширені можливості аналізу.