Kpi-best

Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота № 4

з дисципліни «Функцiйне програмування мовою Haskell»

|  |  |
| --- | --- |
|  | Виконав:  Студент ФІОТ  Групи ІП-93  Владіміров А.О. |

Київ – 2023

**Мета**: Ознайомитись з модульною органiзацiєю програм та засобами введення-виведення. Набути досвiду створення мультимодульних Haskell проєктiв з використанням iнструментiв stack або cabal.

За допомогою iнструментiв stack або cabal створити Haskell-проєкт для

розробки консольного застосунку за темою попередньої лабораторної роботи

(ЛР № 3).

**Завдання**: реалiзувати iнтерфейс командного рядка: а) для виконання функцiй, реалiзованих у ЛР № 3 б) для манiпуляцiї даними (CRUD).

Данi зберiгаються або у текстовому файлi (файлах), або у csv-файлi (фай-

лах) — на вибiр студента. Звiт (результат виконання) виводиться а) на екран (якщо не заборонено, тобто не вказана опцiя [-s|--silent]), б) у текстовий файл, в) у html-файл. Iнтерфейс має пiдтримувати набiр непозицiйних параметрiв (прапорцiв) з, наприклад, таким форматом:

programName [DBNAME] [-c|--command COMMANDNAME [ARGS]]

[-l|--log LOGNAME]

[-s|--silent]

[--html FILENAME]

[-h|--help]

**Виконання**: проєкт має стандартну структуру як для любого Stack проєкту на Haskell. Основні файли програми це файл Lib.hs де зберігається модифікований код з ЛР № 3, Main.hs – код виконання. У папці Input зберігається файл Input.txt де зберігаються тестові дані, з яких програма буде зчитувати, парсити фігури і потім працювати з ними.

У Output зберігаються output.txt і output.html – текстовий і html файл відповідно, у які ми будемо вписувати результат виконання нашого інтерфейсу командного рядка.

Лістинг проєкту:  
Lab.hs

**module** Lib

( Point(..),

Vector(..),

Font(..),

Shape (..),

Plane(..),

FigureType(..),

createShape,

readShape,

updateShape,

deleteShape,

areaShape,

typeShape,

boxShape,

containedShape,

moveShape

)

**where**

**import** Data.**Maybe**

**data** Point **=** Point {x **::** **Double**, y **::** **Double**} **deriving** (**Eq**, **Read**, **Show**)

**newtype** Vector **=** Vector Point **deriving** (**Show**)

**data** Font **=** Consolas **|** LucidaConsole **|** SourceCodePro **deriving** (**Eq**, **Show**)

**instance** **Read** Font **where**

**readsPrec** \_ input **=** **case** input **of**

"Consolas" **->** [(Consolas, "")]

"LucidaConsola" **->** [(LucidaConsole, "")]

"SourceCodePro" **->** [(SourceCodePro, "")]

\_ **->** []

fontArea **::** Font **->** **Double**

fontArea Consolas **=** 1.0

fontArea LucidaConsole **=** 1.2

fontArea SourceCodePro **=** 1.5

**data** Shape

**=** Circle Point **Double**

**|** Rectangle Point Point

**|** Triangle Point Point Point

**|** Label Point Font **String**

**deriving** (**Eq**, **Read**, **Show**)

*-- 1.1 обчислення площi фiгури*

**class** Area a **where**

area **::** a **->** **Double**

**instance** Area Shape **where**

area (Circle \_ r) **=** **pi** **\*** r **^** 2

area (Rectangle (Point x1 y1) (Point x2 y2)) **=** **abs** (x2 **-** x1) **\*** **abs** (y2 **-** y1)

area (Triangle (Point x1 y1) (Point x2 y2) (Point x3 y3)) **=** 0.5 **\*** **abs** ((x1 **-** x3) **\*** (y2 **-** y1) **-** (x1 **-** x2) **\*** (y3 **-** y1))

area (Label (Point x y) font text) **=** **fromIntegral** (**length** text) **\*** fontArea font

*-- 1.2 отримання списку фігур вказаного типу*

**data** FigureType **=** CircleType **|** RectangleType **|** TriangleType **|** LabelType **deriving** (**Eq**, **Show**)

**instance** **Read** FigureType **where**

**readsPrec** \_ input **=** **case** input **of**

"CircleType" **->** [(CircleType, "")]

"RectangleType" **->** [(RectangleType, "")]

"TriangleType" **->** [(TriangleType, "")]

"LabelType" **->** [(LabelType, "")]

\_ **->** []

**class** Figure a **where**

figureType **::** a **->** FigureType

**instance** Figure Shape **where**

figureType (Circle {}) **=** CircleType

figureType (Rectangle {}) **=** RectangleType

figureType (Triangle {}) **=** TriangleType

figureType (Label {}) **=** LabelType

*-- 1.3 отримання прямокутника, що охоплює вказану фiгуру*

**class** BoundingBox a **where**

boundingBox **::** a **->** Shape

**instance** BoundingBox Shape **where**

boundingBox (Circle (Point x y) r) **=** Rectangle (Point (x **-** r) (y **-** r)) (Point (x **+** r) (y **+** r))

boundingBox (Rectangle p1 p2) **=** Rectangle (Point (**min** (x p1) (x p2)) (**min** (y p1) (y p2))) (Point (**max** (x p1) (x p2)) (**max** (y p1) (y p2)))

boundingBox (Triangle p1 p2 p3) **=** Rectangle (Point (**minimum** [x p1, x p2, x p3]) (**minimum** [y p1, y p2, y p3])) (Point (**maximum** [x p1, x p2, x p3]) (**maximum** [y p1, y p2, y p3]))

boundingBox (Label (Point x y) font text) **=**

**let** w **=** **fromIntegral** (**length** text) **\*** 1.0

h **=** fontArea font

**in** Rectangle (Point x y) (Point (x **+** w) (y **+** h))

*-- 1.4 пошук фiгур, якi знаходяться у вказаному квадратi на площинi*

containedIn **::** Shape **->** Shape **->** **Bool**

containedIn (Rectangle (Point x1 y1) (Point x2 y2)) (Rectangle (Point x3 y3) (Point x4 y4)) **=**

**min** x1 x2 **>=** **min** x3 x4 **&&** **max** x1 x2 **<=** **max** x3 x4 **&&** **min** y1 y2 **>=** **min** y3 y4 **&&** **max** y1 y2 **<=** **max** y3 y4

**instance** **Ord** Shape **where**

**compare** s1 s2 **=**

**let** bb1 **=** boundingBox s1

bb2 **=** boundingBox s2

**in** **if** bb1 `containedIn` bb2

**then** LT

**else** GT

*-- 1.5 перемiщення фiгури на вказаний вектор*

**class** Move a **where**

move **::** Vector **->** a **->** a

**instance** Move Shape **where**

move (Vector (Point dx dy)) (Circle (Point x y) r) **=** Circle (Point (x **+** dx) (y **+** dy)) r

move (Vector (Point dx dy)) (Rectangle (Point x1 y1) (Point x2 y2)) **=**

Rectangle (Point (x1 **+** dx) (y1 **+** dy)) (Point (x2 **+** dx) (y2 **+** dy))

move (Vector (Point dx dy)) (Triangle (Point x1 y1) (Point x2 y2) (Point x3 y3)) **=**

Triangle (Point (x1 **+** dx) (y1 **+** dy)) (Point (x2 **+** dx) (y2 **+** dy)) (Point (x3 **+** dx) (y3 **+** dy))

move (Vector (Point dx dy)) (Label (Point x y) font text) **=** Label (Point (x **+** dx) (y **+** dy)) font text

Функції-обгортки над реалізованими функціями з ЛР № 3 + CRUD функції, які працюють з типом Maybe Shape.

**type** Plane **=** [**Maybe** Shape]

*--CRUD*

*-- Create*

createShape **::** **Maybe** Shape **->** Plane **->** Plane

createShape Nothing p **=** p

createShape (Just s) p **=** Just s **:** p

*-- Read*

readShape **::** **Int** **->** Plane **->** **Maybe** Shape

readShape i p

**|** i **<** 0 **||** i **>=** **length** p **=** Nothing

**|** **otherwise** **=** p **!!** i

*-- Update*

updateShape **::** **Int** **->** **Maybe** Shape **->** Plane **->** Plane

updateShape i Nothing p **=** p

updateShape i (Just s) p

**|** i **<** 0 **||** i **>=** **length** p **=** p

**|** **otherwise** **=** **take** i p **++** [Just s] **++** **drop** (i **+** 1) p

*-- Delete*

deleteShape **::** **Int** **->** Plane **->** Plane

deleteShape i p

**|** i **<** 0 **||** i **>=** **length** p **=** p

**|** **otherwise** **=** **take** i p **++** **drop** (i **+** 1) p

*-- functions from lab 3*

areaShape **::** **Int** **->** Plane **->** **Maybe** **Double**

areaShape i p

**|** i **<** 0 **||** i **>=** **length** p **||** isNothing (p **!!** i) **=** Nothing

**|** **otherwise** **=** area **<$>** p **!!** i

typeShape **::** FigureType **->** Plane **->** Plane

typeShape f **=** **filter** (**maybe** False (**\** s **->** figureType s **==** f))

boxShape **::** **Int** **->** Plane **->** **Maybe** Shape

boxShape i p

**|** i **<** 0 **||** i **>=** **length** p **||** isNothing (p **!!** i) **=** Nothing

**|** **otherwise** **=** boundingBox **<$>** p **!!** i

containedShape **::** **Maybe** Shape **->** Plane **->** Plane

containedShape Nothing p **=** []

containedShape (Just r) p **=** **filter** (**maybe** False (**<**r)) p

moveShape **::** **Int** **->** **Maybe** Vector **->** Plane **->** Plane

moveShape \_ Nothing p **=** p

moveShape i (Just v) p **=** **foldr** (**\**x acc **->** **if** **length** acc **==** i **-** 1 **then** (move v **<$>** x) **:** acc **else** x **:** acc) [] p

Main.hs

**module** Main (main) **where**

**import** Lib

**import** System.Exit (exitSuccess)

**import** System.Environment

**import** System.**IO**

**import** Text.**Read** (readMaybe)

**import** Control.**Monad** (unless, when)

**import** Control.Exception (evaluate)

**import** Data.List (tails, isInfixOf, findIndex, isPrefixOf)

**import** **qualified** Data.ByteString.Char8 **as** BS

**import** Data.**Maybe** (fromMaybe)

**import** **Prelude** **hiding** (**log**)

Структура, в якій ми будемо зберігати пропарсену команду.

*-- свій парсер командого рядка*

**data** Command

**=** Create Shape

**|** **Read** **Int**

**|** Update **Int** Shape

**|** Delete **Int**

**|** Area **Int**

**|** Type FigureType

**|** Box **Int**

**|** Contained Shape

**|** Move **Int** Vector

**deriving** (**Show**)

Структура, в якій ми будемо зберігати весь пропарсений рядок.

**data** Options **=** Options

{ db **::** **Maybe** **String**,

command **::** **Maybe** Command,

**log** **::** **Maybe** FilePath,

silent **::** **Bool**,

html **::** **Maybe** FilePath,

help **::** **Bool**

}

**deriving** (**Show**)

Функція, яка використовує Pattern Matching для парсингу фігури з командного рядка.

parseShape **::** [**String**] **->** **Maybe** Shape

parseShape ("rectangle" **:** x1 **:** y1 **:** x2 **:** y2 **:** \_) **=** Just **$** Rectangle (Point (**read** x1) (**read** y1)) (Point (**read** x2) (**read** y2))

parseShape ("circle" **:** x **:** y **:** r **:** \_) **=** Just **$** Circle (Point (**read** x) (**read** y)) (**read** r)

parseShape ("triangle" **:** x1 **:** y1 **:** x2 **:** y2 **:** x3 **:** y3 **:** \_) **=** Just **$** Triangle (Point (**read** x1) (**read** y1)) (Point (**read** x2) (**read** y2)) (Point (**read** x3) (**read** y3))

parseShape ("label" **:** x **:** y **:** font **:** text **:** \_) **=** Just **$** Label (Point (**read** x) (**read** y)) (**read** font) text

parseShape \_ **=** Nothing

Функція, яка використовує Pattern Matching для парсингу вектору з командного рядка.

parseVector **::** [**String**] **->** **Maybe** Vector

parseVector ("vector" **:** dx **:** dy **:** \_) **=** Just **$** Vector (Point (**read** dx) (**read** dy))

Функція, яка використовує Pattern Matching для парсингу команди з командного рядка. Спочатку виконується parseShape і її результат застосовується як аргумент до Create / Read / Update / Delete / Area / Type / Box / Contatined / Move.

parseCommand **::** [**String**] **->** **Maybe** Command

parseCommand ("create" **:** xs) **=** Create **<$>** parseShape xs

parseCommand ("read" **:** i **:** \_) **=** Just **$** **Read** (**read** i)

parseCommand ("update" **:** i **:** xs) **=** Update **<$>** Just (**read** i) **<\*>** parseShape xs

parseCommand ("delete" **:** i **:** \_) **=** Just **$** Delete (**read** i)

parseCommand ("area" **:** i **:** \_) **=** Just **$** Area (**read** i)

parseCommand ("type" **:** t **:** \_) **=** Just **$** Type (**read** t)

parseCommand ("box" **:** i **:** \_) **=** Just **$** Box (**read** i)

parseCommand ("contained" **:** xs) **=** Contained **<$>** parseShape xs

parseCommand ("move" **:** i **:** xs) **=** Move **<$>** Just (**read** i) **<\*>** parseVector xs

parseCommand \_ **=** Nothing

Функція, яка парсить командний рядок у зручну для нас структуру.

parseOptions **::** [**String**] **->** Options

parseOptions args **=**

Options

{ db **=** getParamValue "-db" args,

command **=** parseCommand commandArgs,

**log** **=** getParamValue "-log" args,

silent **=** isParamSet "-silent" args,

html **=** getParamValue "-html" args,

help **=** isParamSet "-help" args

}

**where**

commandArgs **=** **tail** **$** **dropWhile** (**/=** "-command") args

Шукає значення прапорців –html, -log, -db.

getParamValue **::** **String** **->** [**String**] **->** **Maybe** **String**

getParamValue \_ [] **=** Nothing

getParamValue param (x **:** xs) **=** **if** x **==** param **then** Just **$** **head** xs **else** getParamValue param xs

Перевіряє, чи стоїть прапорець –help або –silent.

isParamSet **::** **String** **->** [**String**] **->** **Bool**

isParamSet param args **=** param `**elem**` args

Повертає результат виконання введеної команди.

execCommand **::** **Maybe** Command **->** Plane **->** **IO** (**Maybe** **String**)

*-- CRUD*

execCommand (Just (Create s)) p **=** **do**

**return** **$** Just **$** **show** **$** createShape (Just s) p

execCommand (Just (**Read** i)) p **=** **do**

**return** **$** Just **$** **show** **$** readShape i p

execCommand (Just (Update i s)) p **=** **do**

**return** **$** Just **$** **show** **$** updateShape i (Just s) p

execCommand (Just (Delete i)) p **=** **do**

**return** **$** Just **$** **show** **$** deleteShape i p

*-- area*

execCommand (Just (Area i)) p **=** **do**

**return** **$** Just **$** **show** **$** areaShape i p

*-- type*

execCommand (Just (Type t)) p **=** **do**

**return** **$** Just **$** **show** **$** typeShape t p

*-- bounding box*

execCommand (Just (Box i)) p **=** **do**

**return** **$** Just **$** **show** **$** boxShape i p

*-- contained*

execCommand (Just (Contained r)) p **=** **do**

**return** **$** Just **$** **show** **$** containedShape (Just r) p

*-- move*

execCommand (Just (Move i v)) p **=** **do**

**return** **$** Just **$** **show** **$** moveShape i (Just v) p

execCommand Nothing \_ **=** **return** Nothing

Функція, що пише інструкцію з використання програми.

printHelp **::** **IO** ()

printHelp **=** **do**

**putStrLn** "Usage:"

**putStrLn** " [executable] [options]"

**putStrLn** ""

**putStrLn** "Options:"

**putStrLn** " -db [path] Set path to database file"

**putStrLn** " -command [COMMAND] The command to execute"

**putStrLn** " -log [path] Set path to log file"

**putStrLn** " -silent Run in silent mode"

**putStrLn** " -html [path] Set path to HTML file"

**putStrLn** " -help Print this help message"

**putStrLn** ""

**putStrLn** "Commands:"

**putStrLn** " create rectangle [x1] [y1] [x2] [y2]"

**putStrLn** " create circle [x] [y] [r]"

**putStrLn** " create triangle [x1] [y1] [x2] [y2] [x3] [y3]"

**putStrLn** " create label [x] [y] [font size] [text]"

**putStrLn** " read [id]"

**putStrLn** " update [id] rectangle [x1] [y1] [x2] [y2]"

**putStrLn** " update [id] circle [x] [y] [r]"

**putStrLn** " update [id] triangle [x1] [y1] [x2] [y2] [x3] [y3]"

**putStrLn** " update [id] label [x] [y] [font] [text]"

**putStrLn** " delete [id]"

**putStrLn** " area [id]"

**putStrLn** " type [type]"

**putStrLn** " box [id]"

**putStrLn** " contained rectangle [x1] [y1] [x2] [y2]"

**putStrLn** " contained circle [x] [y] [r]"

**putStrLn** " contained triangle [x1] [y1] [x2] [y2] [x3] [y3]"

**putStrLn** " contained label [x] [y] [font] [text]"

**putStrLn** " move [id] vector [dx] [dy]"

Функція, що парсить файл з вхідними даними.

parseFile **::** **Maybe** FilePath **->** **IO** Plane

parseFile Nothing **=** **putStrLn** "File path not provided" **>>** pure []

parseFile (Just filePath) **=** **do**

contents **<-** **readFile** filePath

**let** parsedPlane **=** readMaybe contents **::** **Maybe** Plane

**case** parsedPlane **of**

Just plane **->** **return** plane

Nothing **->** **error** "Failed to parse file into Plane"

Функція, що записує результат в файл.

appendFileWithContent **::** **String** **->** FilePath **->** **IO** ()

appendFileWithContent content filePath **=** **appendFile** filePath content

maybeAppendFile **::** **Maybe** FilePath **->** **String** **->** **IO** ()

maybeAppendFile maybeFilePath content **=** **maybe** (**return** ()) (appendFileWithContent content) maybeFilePath

Функція, що повністю переписує зміст файлу (для маніпуляцій з вхідним файлом).

maybeRewriteFile **::** **Maybe** FilePath **->** **String** **->** **IO** ()

maybeRewriteFile (Just file) content **=** **writeFile** file content

maybeRewriteFile Nothing \_ **=** **return** ()

Функція, що записує результат в html – файл. Використовує ByteString для strict evaluation.

maybeAppendHtml **::** **Maybe** FilePath **->** **String** **->** **IO** ()

maybeAppendHtml (Just file) result **=** **do**

htmlBytes **<-** BS.**readFile** file

**let** html **=** BS.unpack htmlBytes

modifiedHtml **=** **unlines** **$** **take** 4 (**lines** html) **++** ["**\t\t\t**<li>" **++** result **++** "</li>"] **++** **drop** 4 (**lines** html)

BS.**writeFile** file **$** BS.pack modifiedHtml

maybeAppendHtml Nothing \_ **=** **return** ()

main **::** **IO** ()

main **=** **do**

Зчитуємо командний рядок, парсимо. Якщо є прапорець –help, виводимо інструкцію і виходимо з програми.

args **<-** getArgs

**let** options **=** parseOptions args

when (help options) printHelp **>>** exitSuccess

shapes **<-** parseFile (db options)

maybeResult **<-** execCommand (command options) shapes

**case** maybeResult **of**

Just result **->** **do**

Якщо стоїть пропорець –silent, результат виконання програми нікуди не виводимо і в ніякий файл не зберігаємо.

unless (silent options) **$** **putStrLn** result

unless (silent options) **$** maybeAppendFile (**log** options) (result **++** "**\n**")

unless (silent options) **$** maybeAppendHtml (html options) result

Якщо команда змінує нашу БД, результат виконання програми записуємо у вхідний файл не залишаючи попереднього змісту.

**case** command options **of**

Just (Create \_) **->** maybeRewriteFile (db options) result

Just (Update {}) **->** maybeRewriteFile (db options) result

Just (Delete \_) **->** maybeRewriteFile (db options) result

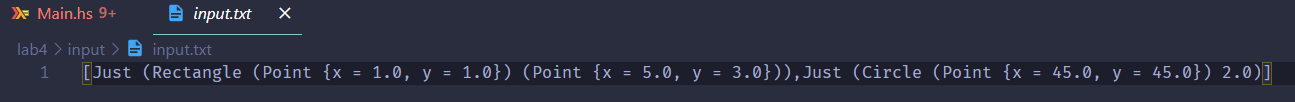
Just (Move {}) **->** maybeRewriteFile (db options) result

\_ **->** **return** ()

Nothing **->** **return** ()

**Приклад запуску програми:**

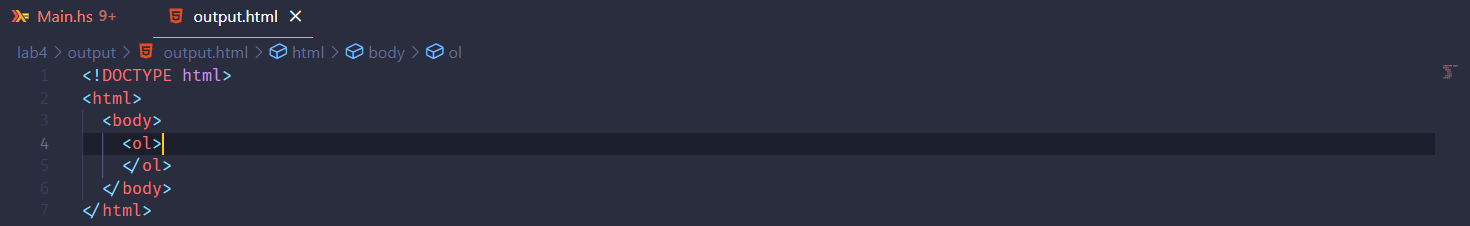
Вхідний файл зберігає наступні дані



Output.txt



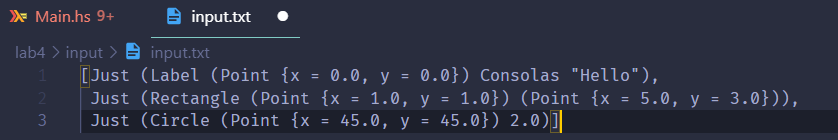
Output.html



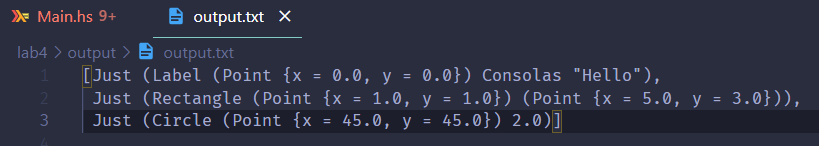
Запускаємо команду

PS C:\Users\User\Documents\haskell\lab4> stack exec lab4-exe -- -db C:\Users\User\Documents\haskell\lab4\input\input.txt -command create label 0 0 Consolas "Hello" -log C:\Users\User\Documents\haskell\lab4\output\output.txt -html C:\Users\User\Documents\haskell\lab4\output\output.html

Input.txt після виконання команди



Output.txt після виконання команди



Output.html після виконання команди

