Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение высшего образования
«Иркутский государственный университет»
(ФГБОУ ВО «ИГУ»)
Институт математики и информационных технологий Кафедра вычислительной математики и оптимизации

КУРСОВАЯ РАБОТА

по направлению «Прикладная математика и информатика»
профиль «Математические методы и информационные технологии»

Операции над матрицами на языке Haskell

Студента 3 курса очного отделения
группы 02321-ДБ
Скрыпникова Никиты Юрьевича

Руководитель:
к. ф.-м. н., доцент
_____ Черкашин Е. А.

Иркутск - 2022

Оглавление

Введение

Haskell – функциональный язык программирования, сильно отличающийся от императивных и смешанных языков разработки. Важной особенность языка является поддержка ленивых вычислений, что позволяет ускорить работу программы. В Haskell аргументы функции вычисляются только тогда, когда действительно требуются для вычисления. Причем ленивые вычисления выполняются без вмешательства самого программиста.

Haskell – актуален. Язык применяется в финансовом секторе – для разработки собственных инструментов в крупных банках и других компаниях. Также Haskell часто используется для обработки текстов, синтаксического анализа, а также веб-разработки.

Целью данной курсовой работы является ознакомление с Haskell'ем посредством решения задач, связанных с графами, укрепление и развитие навыков программирования.

Цель курсовой работы заключается в разработке приложения «Сложение и вычитание матриц в файле» на языке функционального программирования.

Теоретическая часть

Матрица - математический объект, записываемый в виде прямоугольной таблицы, которая представляет собой совокупность строк и столбцов, на пересечении которых находятся её элементы. Количество строк и столбцов матрицы задают размер матрицы. Хотя исторически рассматривались, например, треугольные матрицы, в настоящее время говорят исключительно о матрицах прямоугольной формы, так как они являются наиболее удобными и общими.

Матрицы широко применяются в математике для компактной записи систем линейных алгебраических или дифференциальных уравнений. В этом случае, количество строк матрицы соответствует числу уравнений, а количество столбцов - количеству неизвестных. В результате решение систем линейных уравнений сводится к операциям над матрицами.

Для матрицы определены следующие алгебраические операции:

- сложение и вычитание матриц, имеющих один и тот же размер;
- умножение матриц подходящего размера, в том числе умножение матриц на вектор;
- умножение матрицы на число.

Сложение и вычитание матриц

Пусть матрицы *A* и *B* состоят из одинакового числа строк и одинакового числа столбцов, т.е. имеют одинаковые размеры. Тогда для того, чтобы сложить матрицы *A* и *B* нужно к элементам матрицы *A* прибавить элементы матрицы *B*, стоящие на тех же местах. Таким образом, суммой двух матриц *A* и *B* называется матрица *C*, которая определяется по правилу, например,

$$A+B=\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}+\begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix}=\begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} \\ a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} \end{pmatrix}$$

или

$$(c_{ij})=(a_{ij}+b_{ij})$$

С вычитанием действуем аналогичным образом.

Умножение матрицы на число

Для того чтобы умножить матрицу *A* на число *k* нужно каждый элемент матрицы *A* умножить на это число. Таким образом, произведение матрицы *A* на число *k* есть новая матрица, которая определяется по правилу $kA=k\cdot\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix}=\begin{pmatrix} ka_{11} & ka_{12} \\ ka_{21} & ka_{22} \\ ka_{31} & ka_{32} \end{pmatrix}$ или $(c_{ij})=(ka_{ij})$.

Умножение матриц

Эта операция осуществляется по своеобразному закону. Прежде всего, заметим, что размеры матриц–сомножителей должны быть согласованы. Перемножать можно только те матрицы, у которых число столбцов первой матрицы совпадает с числом строк второй матрицы (т.е. длина строки первой равна высоте столбца второй). *Произведением* матрицы *A* на матрицу *B* называется новая матрица *C=AB*, элементы которой составляются следующим образом:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}\cdot\begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix}=\begin{pmatrix} a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22} & a_{11}b_{13}+a_{12}b_{23} \\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22} & a_{21}b_{13}+a_{22}b_{23} \end{pmatrix}.$$

Таким образом, например, чтобы получить у произведения (т.е. в матрице *C*) элемент, стоящий в 1-ой строке и 3-м столбце c_{13}, нужно в 1-ой матрице взять 1-ую строку, во 2-ой – 3-й столбец, и затем элементы строки умножить на соответствующие элементы столбца и полученные произведения сложить. И другие элементы матрицы-произведения получаются с помощью аналогичного произведения строк первой матрицы на столбцы второй матрицы.

В общем случае, если мы умножаем матрицу $A = (a_{ij})$ размера $m \times n$ на матрицу $B = (b_{ij})$ размера $n \times p$, то получим матрицу C размера $m \times p$, элементы которой вычисляются следующим образом: элемент c_{ij} получается в результате произведения элементов i-ой строки матрицы A на соответствующие элементы j-го столбца матрицы B и их сложения.

Для решения задачи данной курсовой работы был выбран функциональный язык программирования Haskell. Важной особенностью Haskell является то, что он поддерживает ленивые вычисления, которые позволяют сильно ускорить работу программы и снизить нагрузку на память, а также сделать код проще и модульнее. Ленивые вычисления выполняются тогда, когда это необходимо программе, а не в том моменте, когда их указал разработчик.

В обычных — не ленивых или «энергичных» вычислениях — аргументы функции вычисляются перед выполнением самой функции. В языках программирования, которые применяют ленивые вычисления, этот процесс постоянно откладывается, и аргументы функции вычисляются только для реальной надобности, а не в том месте, где их указал разработчик. Например, если сейчас значение какой-то функций не нужно, и оно не используется, то Haskell не будет высчитывать ее аргументы.

Реализация

Разработанная программа представляет собой консольное приложение, которое выводит данные, считанные из текстовых файлов A.txt и B.txt, и результат вычислений (операции сложения и вычитания).

Также реализована дополнительная функция умножения матриц на вектор.

```
module Main where
import System.IO
import Prelude

main :: IO ()
main = do

 fileSourceA <- readFile "A.txt"
 fileSourceB <- readFile "B.txt"
 fileSourceV <- readFile "Vector.txt"

 let matrixA = map (\x ->  map (\i -> (read i) :: Int) $ words x) $ lines fileSourceA
 let matrixB = map (\x ->  map (\i -> (read i) :: Int) $ words x) $ lines fileSourceB
```

```
let vector = (map (\x ->  (read x) :: Int) $ words fileSourceV)
let matrixC = zipWith (\x y -> zipWith (\i j -> (i + j)) x y) matrixA matrixB
let matrixD = zipWith (\x y -> zipWith (\i j -> (i - j)) x y) matrixA matrixB
let vectorA = map (\x -> sum x) (map (\x  -> zipWith (*) x vector) matrixA)
let vectorB = map (\x -> sum x) (map (\x  -> zipWith (*) x vector) matrixB)


print matrixA
print matrixB
print vector
print matrixC
print matrixD
print vectorA
print vectorB
return ()
```

Листинг - 1

Данные для матрицы и вектора считываются из соответствующих файлов A.txt и Vector.txt.

Используемые библиотеки и функции:

- System.IO - стандартная библиотека ввода-вывода.
- Prelude - стандартный модуль. Prelude по умолчанию импортируется во все модули Haskell, если только для него не указан явный оператор импорта или не включено расширение NoImplicitPrelude.
- map - возвращает список, созданный путем применения функции (первый аргумент) ко всем элементам в списке, переданном в качестве второго аргумента.
- read - приводит строку к другому типу.
- words - создает массив строк из исходного, пробельные символы служат разделителями.
- lines - создает массив строк из исходного, символы новой строки служат разделителями.
- zipWith - составляет список, его элементы вычисляются из функции и элементов входных списков, находящихся на одной и той же позиции в обоих списках.
- readFile - читает файл и возвращает содержимое файла в виде строки.
- sum - вычисляет сумму всех элементов в списке.

- print – печатает данные.

Алгоритм программы

1. Программа считывает данные из текстового файла A.txt, B.txt, Vector.txt, используя функцию readFile.

Для того чтобы данные файлов представить в виде матриц, использовались функции read, words, lines, map.

2. Вычисляет значения матриц matrixC, matrixD и вектора vectorA, используя операции сложения и вычитания матриц, а также умножения матрицы на вектор.

2.1 Операция сложения матриц:

2.1.1. Складывает значение каждого столбца каждой строки матрицы matrixA с соответствующим значением матрицы matrixB. Для этого использовались функция zipWith и оператор +.

Результатом выполнения операции сложения является матрица matrixC.

2.2. Операция вычитания матриц:

2.2.1. Вычитает значение каждого столбца каждой строки матрицы matrixA из соответствующего значения матрицы matrixB.

Для этого использовались функция zipWith и оператор -.

Результатом выполнения операции вычитания является матрица matrixD.

2.3. Операция умножения матрицы на вектор:

2.3.1. Умножение матрицы на вектор производится по правилу «строка на столбец».

Умножает значение каждого столбца каждой строки матрицы matrixA на соответствующее значение столбца каждой строки вектора vector и суммирует полученные значения. Для этого использовались функция map, zipWith, sum и оператор *.

Результатом выполнения операции умножения матрицы matrixA на вектор vector является вектор vectorA.

Результатом выполнения операции умножения матрицы matrixB на вектор vector является вектор vectorB.

3. Выводит результаты вычислений на экран.

```
ghci> main
[[4,1,3],[1,6,6],[7,8,3]]
[[3,1,1],[11,2,1],[1,5,8]]
[2,3,1]
[[7,2,4],[12,8,7],[8,13,11]]
[[1,0,2],[-10,4,5],[6,3,-5]]
[14,26,41]
[10,29,25]
ghci>
```

Обозначение:

[[4,1,3],[1,6,6],[7,8,3]] – вывод матрицы matrixA

[[3,1,1],[11,2,1],[1,5,8]] – вывод матрицы matrixB

[2,3,1] – вывод вектора vector

[[7,2,4],[12,8,7],[8,13,11]] – вывод матрицы matrixC

[[1,0,2],[-10,4,5],[6,3,-5]] – вывод матрицы matrixD

[14,26,41] – вывод вектора vectorA

[10,29,25] – вывод вектора vector

Заключение

Выполнение курсовой работы позволило закрепить ранее изученный материал по дисциплине «Функциональное программирование», изучить новый для меня материал.

В ходе выполнения данной курсовой работы на основе различных источников данных была проанализирована заданная предметная область.

В результате выполнения проекта была полностью реализована программа «Сложение и вычитание матриц в файле».

Список литературы

1. Миран Липовача Изучай Haskell во имя добра! / Пер. с англ. Леушина Д., Синицына А., Арсанукаева Я.– М.: ДМК Пресс, 2012. – 490 с.
2. Г.М. Сергиевский, Н.Г. Волченков. Функциональное и логическое программирование. – М.: Академия, 2010. – 320 с.
3. Р.В. Душкин. 14 занимательных эссе о языке Haskell и функциональном программировании. – М.: ДМК Пресс, 2016. – 222 с.
4. Кубенский, А. А. Функциональное программирование: учебник и практикум для вузов / А. А. Кубенский. — Москва: Издательство Юрайт, 2022. — 348 с.