

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФГБОУ ВО «АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
КАФЕДРА ВТиЭ

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
по курсу «Практикум по технологиям разработки программного
обеспечения»

Выполнил:
студент 576 группы:
_____ Р. А. Щиголев
«__» _____ 2021 г.

Проверил:
ст. преп. каф. ВТиЭ
_____ П. Н. Уланов
«__» _____ 2021 г.

Барнаул, 2021 г.

Содержание

1	Введение и постановка задачи	2
2	Теоретическое описание задачи	2
3	Алгоритм и блок-схема	4
4	Проверка работы программы	15
	Выводы по работе	19
	Приложение	20
	Вывод git log	20
	UML-диаграмма	21
	Листинг main.py	21
	Листинг classes.py	33
	Листинг parsing.py	40

1 Введение и постановка задачи

В данной лабораторной работе необходимо разработать программу для выполнения различных математических преобразований над скалярными, векторными и тензорными величинами - скалярами, векторами и матрицами в объектно-ориентированном подходе. Интерфейс графический или командно-строчный. Интерфейс должен позволять в процессе выполнения программы задать последовательность выполнения действий: задать величины, рассчитать значения, выдать значения в интерфейс (или в текстовые файлы, или в табличные файлы, на выбор).

Во время разработки должен использоваться Git.

2 Теоретическое описание задачи

Для реализации такой программы будем использовать язык программирования Python.

Для работы с матрицами и тензорами была выбрана библиотека NumPy. NumPy - это библиотека языка Python, добавляющая поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых (и очень быстрых) математических функций для операций с этими массивами.

Интерфейс для данного приложения будет команднострочный. Для этого был выбран модуль Click. Click решает ту же проблему, что и optparse и argparse, но немного иначе. Он использует декораторы, поэтому команды должны быть функциями, которые можно обернуть этими декораторами. Сам интерфейс программы разбит на команды:

- Scalar;
- Vector;
- Matrix.

У каждой команды имеются подкоманды, например: matrix determinant - вычисление детерминанта (определителя) матрицы и т.д.

Скаляр - величина, каждое значение которой может быть выражено одним (действительным) числом. Примерами скалярами являются: длина, площадь, время, масса, плотность, температура, работа и др. Термин скаляр употребляется (иногда просто как синоним числа) в векторном исчислении, где скаляр противопоставляется вектору. Список операций над скалярами, необходимый для реализации:

1. Сумма $c = a + b$;
2. Инверсия $b = -a$;
3. Произведение $c = a * b$;

4. Возведение в степень $c = a^b$;
5. Вычисление корня $c = \sqrt[b]{a}$;
6. Расчет основных тригонометрических функций: $\cos, \sin, \tan, \text{ctg}$.

Вектор - это направленный отрезок, то есть отрезок, имеющий длину и определенное направление. Графически вектора изображаются в виде направленных отрезков прямой определенной длины. Список операций над векторами, необходимый для реализации:

1. Умножение вектора на скаляр $C[i] = b * A[i]$;
2. Поэлементное сложение $C[i] = A[i] + B[i]$;
3. Поэлементное умножение $C[i] = A[i] * B[i]$;
4. Умножение вектора на матрицу $C[j] = \sum_{l=0}^{L-1} A[l] * B[l][j]$;
5. Скалярное произведение $C = \sum_{l=0}^{L-1} A[l]B[l]$;
6. Векторное произведение трехмерных векторов
 $C = [AB] = [A, B] = A \times B = A \wedge B$;
7. Вычисление длины (модуля) вектора $\|A\| = \sqrt{\sum_{l=0}^{L-1} A[l]^2}$;
8. Проверка сонаправленности векторов;
9. Проверка векторов на ортогональность.

Матрица - математический объект, записываемый в виде прямоугольной таблицы элементов кольца или поля (например, целых, действительных или комплексных чисел), который представляет собой совокупность строк и столбцов, на пересечении которых находятся его элементы. Список операций над матрицами, необходимый для реализации:

1. Умножение матрицы на скаляр $C[i][j] = b * A[i][j]$ - нужно каждый элемент матрицы умножить на данный скаляр;
2. Поэлементное сложение $C[i][j] = A[i][j] + B[i][j]$ - все элементы которой равны попарной сумме всех соответствующих элементов матриц A и B;
3. Поэлементное произведение $C[i][j] = A[i][j]B[i][j]$ - результатом поэлементного умножения матриц A и B является матрица, каждый элемент которой представляет собой произведение соответствующих элементов матриц A и B;

4. Умножение вектора на матрицу $C[j] = \sum_{l=0}^{L-1} A[l] * B[l][j]$ - при умножении матрицы на вектор-столбец число столбцов в матрице должно совпадать с числом строк в векторе-столбце;
5. Матричное произведение $C[i][j] = \sum_{l=0}^{L-1} A[i][l]B[l][j]$ - произведением двух матриц A и B называется матрица C , элемент которой, находящийся на пересечении i -й строки и j -го столбца, равен сумме произведений элементов i -й строки матрицы A на соответствующие (по порядку) элементы j -го столбца матрицы B ;
6. Вычисление следа матрицы $Tr(A)$ - это сумма элементов квадратной матрицы, расположенных на главной диагонали;
7. Вычисление определителя матрицы $det(A)$ - это сумма слагаемых всевозможных произведений элементов матрицы, взятых по одному из каждой строки и каждого столбца матрицы, при этом знак произведения определяется четностью перестановки;
8. Вычисление обратной матрицы. Обратной матрицей, к квадратной матрице A , называется такая матрица A^{-1} , для которой справедливо равенство: $A = A^{-1} = A^{-1} * A = E$;
9. Транспонирование $B[i][j] = A[j][i]$ - матрица, полученная из исходной матрицы заменой строк на столбцы.

3 Алгоритм и блок-схема

Алгоритм работы программы:

1. Начало программы;
2. Анализ аргументов командной строки;
3. Проверка условия: Было обнаружено прерывание процесса? Если условие выполняется, то переход к п. 145;
4. Проверка условия: Была ли вызвана команда scalar? Если не выполняется, то переход к п. 51;
5. Проверка условия: Было обнаружено прерывание процесса? Если условие выполняется, то переход к п. 145;
6. Проверка условия: Была ли вызвана подкоманда "Сумма двух скаляров"? Если условие не выполняется, то переход к п. 11;
7. Ввод скаляров;

8. Проверка условия: Это число? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
9. Сложение двух скаляров;
10. Вывод результата и переход к п. 145;
11. Проверка условия: Была ли вызвана подкоманда "Инверсия скаляра"? Если условие не выполняется, то переход к п. 16;
12. Ввод скаляра;
13. Проверка условия: Это число? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
14. Инверсия скаляра;
15. Вывод результата и переход к п. 145;
16. Проверка условия: Была ли вызвана подкоманда "Произведение двух скаляров"? Если условие не выполняется, то переход к п. 21;
17. Ввод скаляров;
18. Проверка условия: Это число? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
19. Произведение двух скаляров;
20. Вывод результата и переход к п. 145;
21. Проверка условия: Была ли вызвана подкоманда "Возведение в степень"? Если условие не выполняется, то переход к п. 26;
22. Ввод скаляра и числа;
23. Проверка условия: Это числа? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
24. Возведение скаляра в степень;
25. Вывод результата и переход к п. 145;
26. Проверка условия: Была ли вызвана подкоманда "Вычисление корня"? Если условие не выполняется, то переход к п. 31;
27. Ввод скаляра и степени;
28. Проверка условия: Это числа? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;

29. Взятие корня нужной степени из скаляра;
30. Вывод результат и переход к п. 145;
31. Проверка условия: Была ли вызвана подкоманда "Синус"? Если условие не выполняется, то переход к п. 36;
32. Ввод скаляра;
33. Проверка условия: Это число? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
34. Получение синуса;
35. Вывод результата и переход к п. 145;
36. Проверка условия: Была ли вызвана подкоманда "Косинус"? Если условие не выполняется, то переход к п. 41;
37. Ввод скаляра;
38. Проверка условия: Это число? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
39. Получение косинуса;
40. Вывод результата и переход к п. 145;
41. Проверка условия: Была ли вызвана подкоманда "Тангенс"? Если условие не выполняется, то переход к п. 46;
42. Ввод скаляра;
43. Проверка условия: Это число? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
44. Получение тангенса;
45. Вывод результата и переход к п. 145;
46. Проверка условия: Была ли вызвана подкоманда "Котангенс"? Если условие не выполняется, то переход к п. 5;
47. Ввод скаляра;
48. Проверка условия: Это число? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
49. Получение котангенса;
50. Вывод результата и переход к п. 145;

51. Проверка условия: Была ли вызвана команда `vector`? Если условие не выполняется, то переход к п. 98;
52. Проверка условия: Было обнаружено прерывание процесса? Если условие выполняется, то переход к п. 145;
53. Проверка условия: Была ли вызвана подкоманда "Умножение вектора на скаляр"? Если условие не выполняется, то переход к п. 58;
54. Ввод вектора и скаляра;
55. Проверка условия: Успешна ли валидация? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
56. Умножения вектора на скаляр;
57. Вывод результата и переход к п. 145;
58. Проверка условия: Была ли вызвана подкоманда "Поэлементное сложение"? Если условие не выполняется, то переход к п. 63;
59. Ввод векторов;
60. Проверка условия: Успешна ли валидация? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
61. Поэлементное сложение векторов;
62. Вывод результата и переход к п. 145;
63. Проверка условия: Была ли вызвана подкоманда "Поэлементное умножение"? Если условие не выполняется, то переход к п. 68;
64. Ввод векторов;
65. Проверка условия: Успешна ли валидация? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
66. Поэлементное умножение векторов;
67. Вывод результата и переход к п. 145;
68. Проверка условия: Была ли вызвана подкоманда "Умножения вектора на матрицу"? Если условие не выполняется, то переход к п. 73;
69. Ввод вектора и число строк, столбцов для матрицы;
70. Проверка условия: Успешна ли валидация? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;

71. Умножение вектора на матрицу;
72. Вывод результата и переход к п. 145;
73. Проверка условия: Была ли вызвана подкоманда "Скалярное произведение"? Если условие не выполняется, то переход к п. 78;
74. Ввод векторов;
75. Проверка условия: Успешна ли валидация? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
76. Нахождение скалярного произведения;
77. Вывод результата и переход к п. 145;
78. Проверка условия: Была ли вызвана подкоманда "Векторное произведение"? Если условие не выполняется, то переход к п. 83;
79. Ввод векторов;
80. Проверка условия: Успешна ли валидация? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
81. Нахождение векторного произведения;
82. Вывод результата и переход к п. 145;
83. Проверка условия: Была ли вызвана подкоманда "Вычисление модуля вектора"? Если условие не выполняется, то переход к п. 88;
84. Ввод вектора;
85. Проверка условия: Успешна ли валидация? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
86. Нахождение модуля вектора;
87. Вывод результата и переход к п. 145;
88. Проверка условия: Была ли вызвана подкоманда "Проверка сонаправленности векторов"? Если условие не выполняется, то переход к п. 93;
89. Ввод вектора;
90. Проверка условия: Успешна ли валидация? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
91. Выполнение проверки на сонаправленность векторов;

92. Вывод результата и переход к п. 145;
93. Проверка условия: Была ли вызвана подкоманда "Проверка векторов на ортогональность"? Если условие не выполняется, то переход к п. 52;
94. Ввод вектора;
95. Проверка условия: Успешна ли валидация? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
96. Выполнение проверки на ортогональность векторов;
97. Вывод результата и переход к п. 145;
98. Проверка условия: Была ли вызвана команда matrix? Если условие не выполняется, то переход к п. 3;
99. Проверка условия: Было обнаружено прерывание процесса? Если условие выполняется, то переход к п. 145;
100. Проверка условия: Была ли вызвана подкоманда "Умножение матрицы на скаляр"? Если условие не выполняется, то переход к п. 105;
101. Ввод скаляра и число строк, столбцов для матрицы;
102. Проверка условия: Успешна ли валидация? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
103. Умножение матрицы на скаляр;
104. Вывод результат и переход к п. 145;
105. Проверка условия: Была ли вызвана подкоманда "Поэлементное сложение"? Если условие не выполнятся, то переход к п. 110;
106. Ввод размерности для первой и второй матрицы;
107. Проверка условия: Успешна ли валидация? Если не выполнятся, то вывод сообщения об ошибке и переход к п. 145;
108. Поэлементное сложение матриц;
109. Вывод результат и переход к п. 145
110. Проверка условия: Была ли вызвана подкоманда "Поэлементное произведение"? Если условие не выполнятся, то переход к п. 115;
111. Ввод размерности для первой и второй матрицы;

112. Проверка условия: Успешна ли валидация? Если не выполняются, то вывод сообщения об ошибке и переход к п. 145;
113. Поэлементное произведение матриц;
114. Вывод результат и переход к п. 145
115. Проверка условия: Была ли вызвана подкоманда "Умножение вектора на матрицу"? Если условие не выполняется, то переход к п. 120;
116. Ввод вектора и размерности для матрицы;
117. Проверка условия: Успешна ли валидация? Если не выполняются, то вывод сообщения об ошибке и переход к п. 145;
118. Умножение вектора на матрицу;
119. Вывод результат и переход к п. 145
120. Проверка условия: Была ли вызвана подкоманда "Матричное произведение"? Если условие не выполняется, то переход к п. 125;
121. Ввод размерности для первой и второй матрицы;
122. Проверка условия: Успешна ли валидация? Если не выполняются, то вывод сообщения об ошибке и переход к п. 145;
123. Умножение матриц;
124. Вывод результат и переход к п. 145
125. Проверка условия: Была ли вызвана подкоманда "Вычисление следа"? Если условие не выполняется, то переход к п. 130;
126. Ввод размерности матрицы;
127. Проверка условия: Успешна ли валидация? Если не выполняются, то вывод сообщения об ошибке и переход к п. 145;
128. Вычисление следа матрицы;
129. Вывод результат и переход к п. 145
130. Проверка условия: Была ли вызвана подкоманда "Вычисление определителя"? Если условие не выполняется, то переход к п. 135;
131. Ввод размерности матрицы;
132. Проверка условия: Успешна ли валидация? Если не выполняются, то вывод сообщения об ошибке и переход к п. 145;

133. Вычисление определителя матрицы;
134. Вывод результат и переход к п. 145
135. Проверка условия: Была ли вызвана подкоманда "Вычисление обратной матрицы"? Если условие не выполняется, то переход к п. 140;
136. Ввод размерности матрицы;
137. Проверка условия: Успешна ли валидация? Если не выполняется, то вывод сообщения об ошибке и переход к п. 145;
138. Вычисление обратной матрицы;
139. Вывод результат и переход к п. 145
140. Проверка условия: Была ли вызвана подкоманда "Транспонирование"? Если условие не выполняется, то переход к п. 99;
141. Ввод размерности матрицы;
142. Проверка условия: Успешна ли валидация? Если не выполняется, то вывод сообщения об ошибке и переход к п. 145;
143. Выполнение транспонирование матрицы;
144. Вывод результат и переход к п. 145
145. Конец программы.

Блок-схема программы:

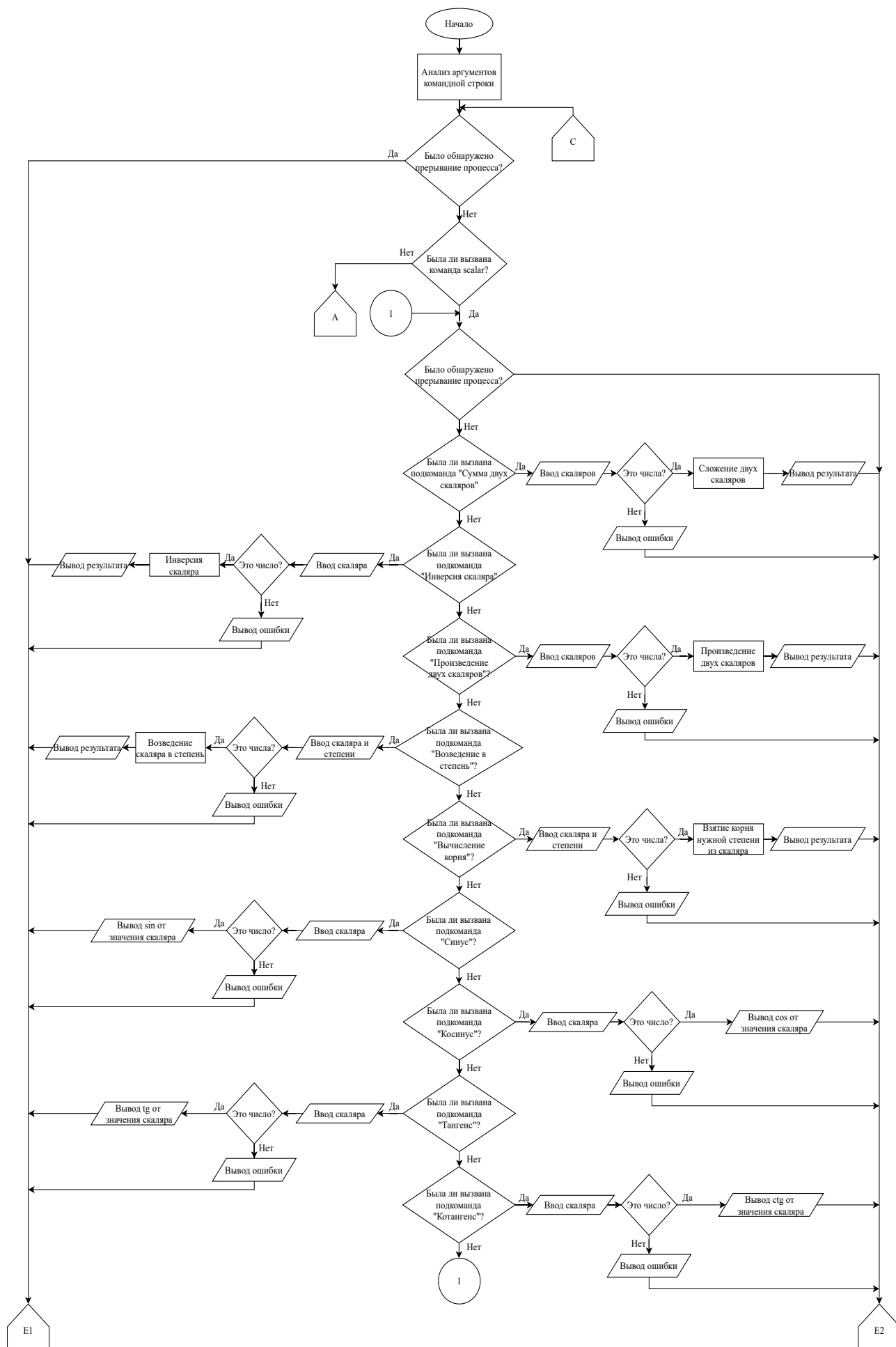


Рис. 1. Блок-схема для команды "Скаляр"

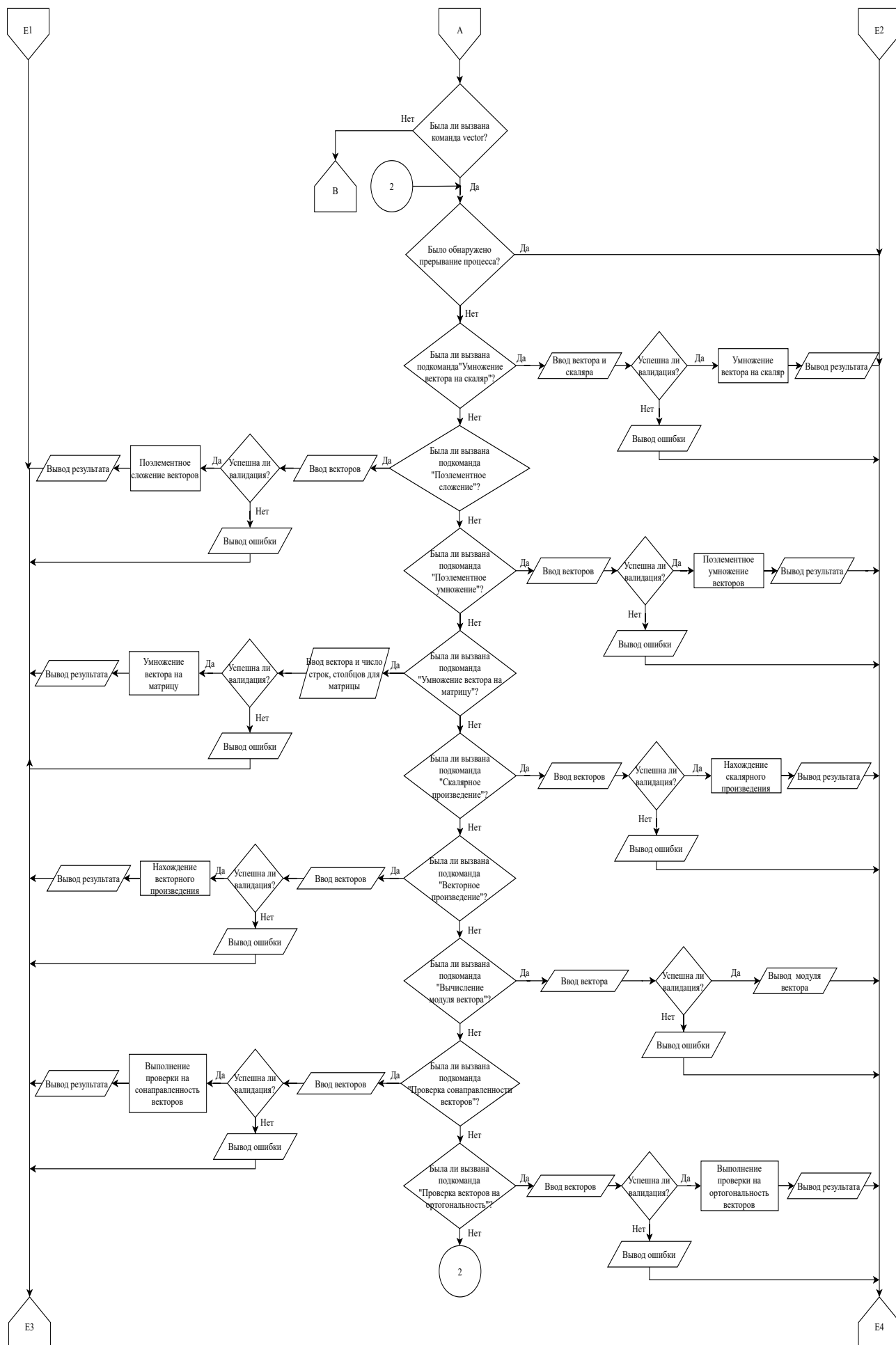


Рис. 2. Блок-схема для команды "Вектор"

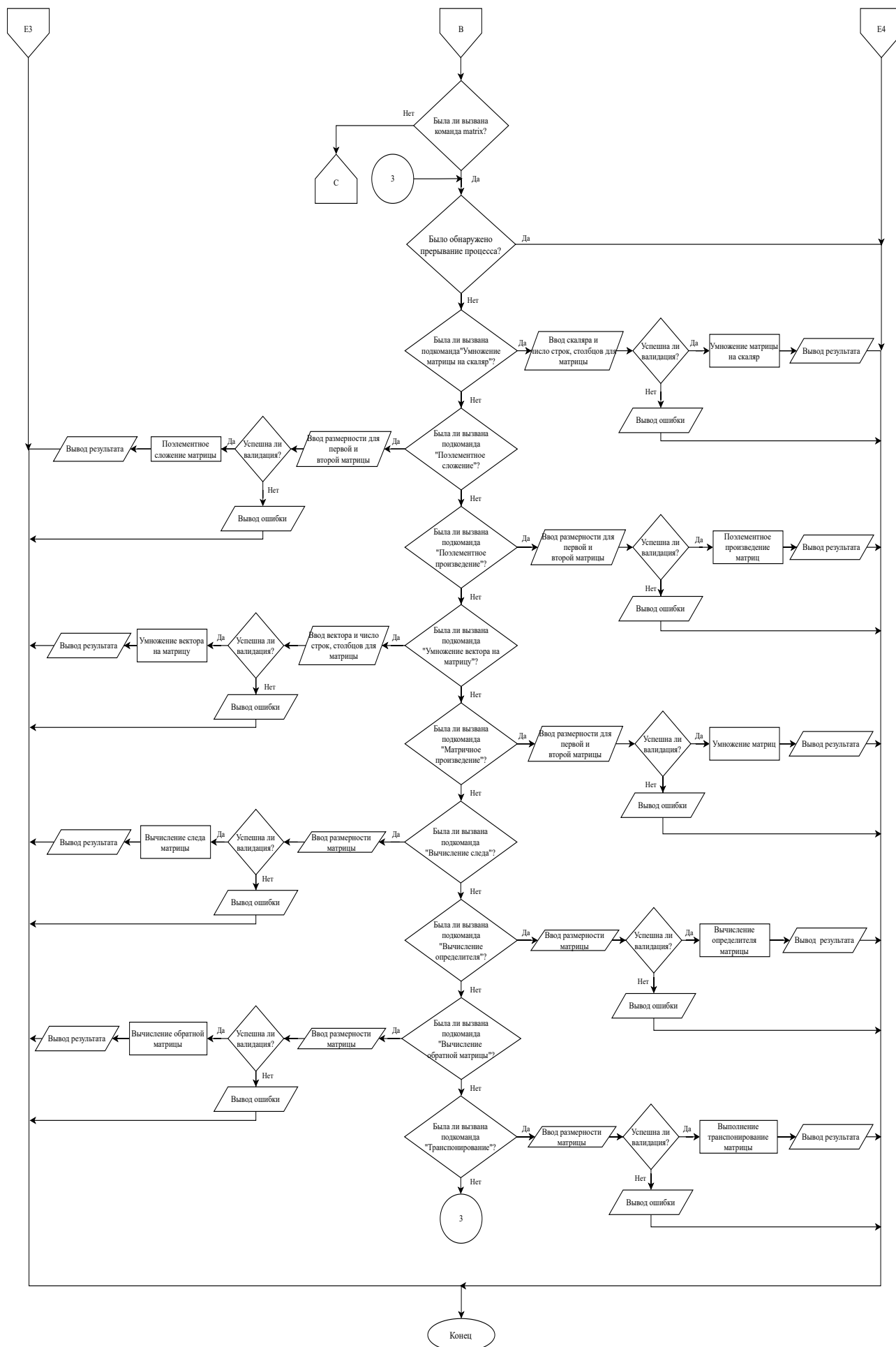
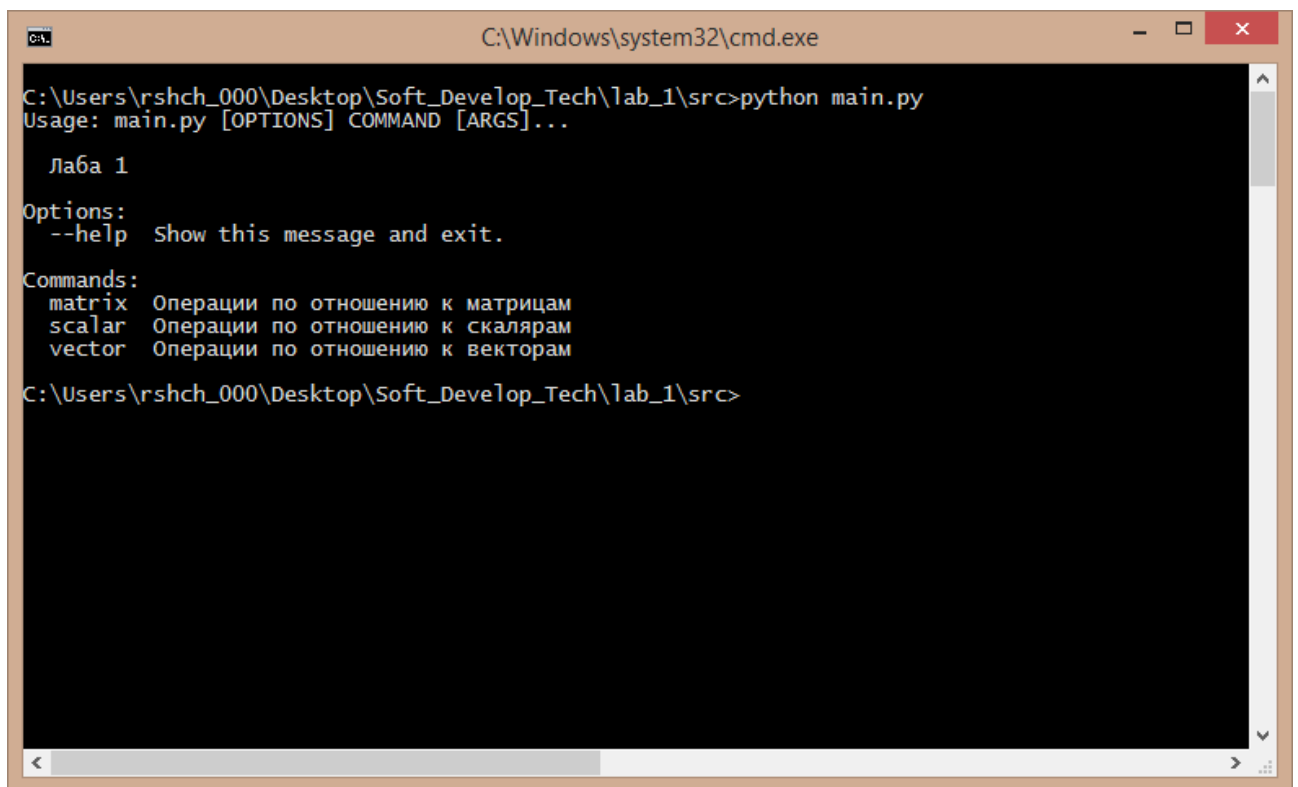


Рис. 3. Блок-схема для команды "Матрица"

4 Проверка работы программы



```
C:\Windows\system32\cmd.exe

C:\Users\rshch_000\Desktop\Soft_Develop_Tech\lab_1\src>python main.py
Usage: main.py [OPTIONS] COMMAND [ARGS]...

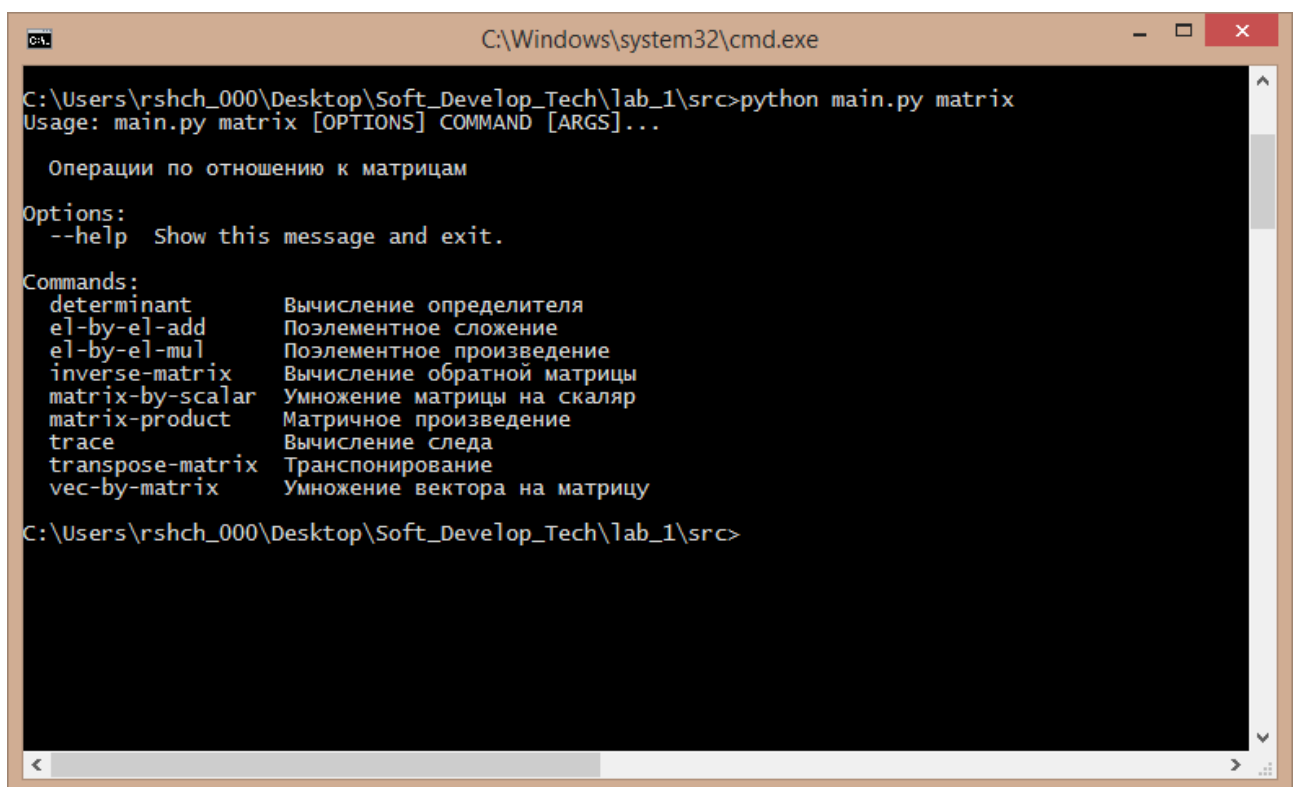
  лаба 1

Options:
  --help  Show this message and exit.

Commands:
  matrix  Операции по отношению к матрицам
  scalar  Операции по отношению к скалярам
  vector  Операции по отношению к векторам

C:\Users\rshch_000\Desktop\Soft_Develop_Tech\lab_1\src>
```

Рис. 4. Команды



```
C:\Windows\system32\cmd.exe

C:\Users\rshch_000\Desktop\Soft_Develop_Tech\lab_1\src>python main.py matrix
Usage: main.py matrix [OPTIONS] COMMAND [ARGS]...

  Операции по отношению к матрицам

Options:
  --help  Show this message and exit.

Commands:
  determinant      Вычисление определителя
  el-by-el-add      Поэлементное сложение
  el-by-el-mul      Поэлементное произведение
  inverse-matrix    Вычисление обратной матрицы
  matrix-by-scalar  Умножение матрицы на скаляр
  matrix-product    Матричное произведение
  trace            Вычисление следа
  transpose-matrix  Транспонирование
  vec-by-matrix     Умножение вектора на матрицу

C:\Users\rshch_000\Desktop\Soft_Develop_Tech\lab_1\src>
```

Рис. 5. Подкоманды для матрицы


```
C:\Windows\system32\cmd.exe
C:\Users\rshch_000\Desktop\Soft_Develop_Tech\lab_1\src>python main.py matrix el-by-el-add
Введите количество строк для первой матрицы: 3
Введите количество столбцов для первой матрицы: 3
Введите количество строк для второй матрицы: 3
Введите количество столбцов для второй матрицы: 3
Введите элементы для 1-ой строки через пробел: 1 2 3
Введите элементы для 2-ой строки через пробел: 4 5 6
Введите элементы для 3-ой строки через пробел: 7 8 9
Введите элементы для 1-ой строки через пробел: 1 1 1
Введите элементы для 2-ой строки через пробел: 2 2 2
Введите элементы для 3-ой строки через пробел: 3 3 3
[[ 2.  3.  4.]
 [ 6.  7.  8.]
 [10. 11. 12.]]
C:\Users\rshch_000\Desktop\Soft_Develop_Tech\lab_1\src>
```

Рис. 6. Поэлементное сложение

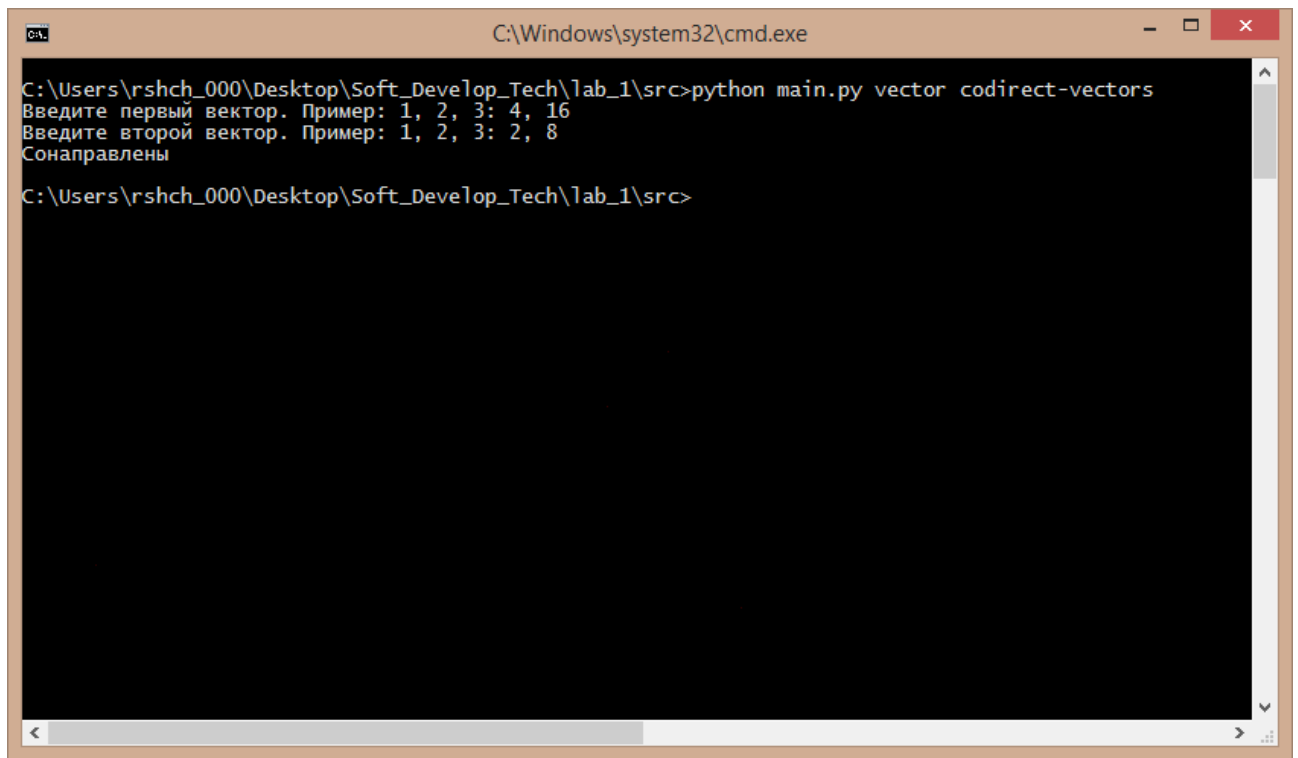
```
C:\Windows\system32\cmd.exe
C:\Users\rshch_000\Desktop\Soft_Develop_Tech\lab_1\src>python main.py vector
Usage: main.py vector [OPTIONS] COMMAND [ARGS]...

Операции по отношению к векторам

Options:
  --help  Show this message and exit.

Commands:
  codirect-vectors  Проверка сонаправленности векторов
  el-by-el-add      Поэлементное сложение
  el-by-el-mul      Поэлементное умножение
  module-vector     Вычисление длины (модуля) вектора
  orthog-vectors    Проверка векторов на ортогональность
  scalar-product    Скалярное произведение
  vec-by-matrix     Умножение вектора на матрицу
  vec-by-scalar     Умножение вектора на скаляр
  vector-product    Векторное произведение
C:\Users\rshch_000\Desktop\Soft_Develop_Tech\lab_1\src>
```

Рис. 7. Подкоманды для вектора

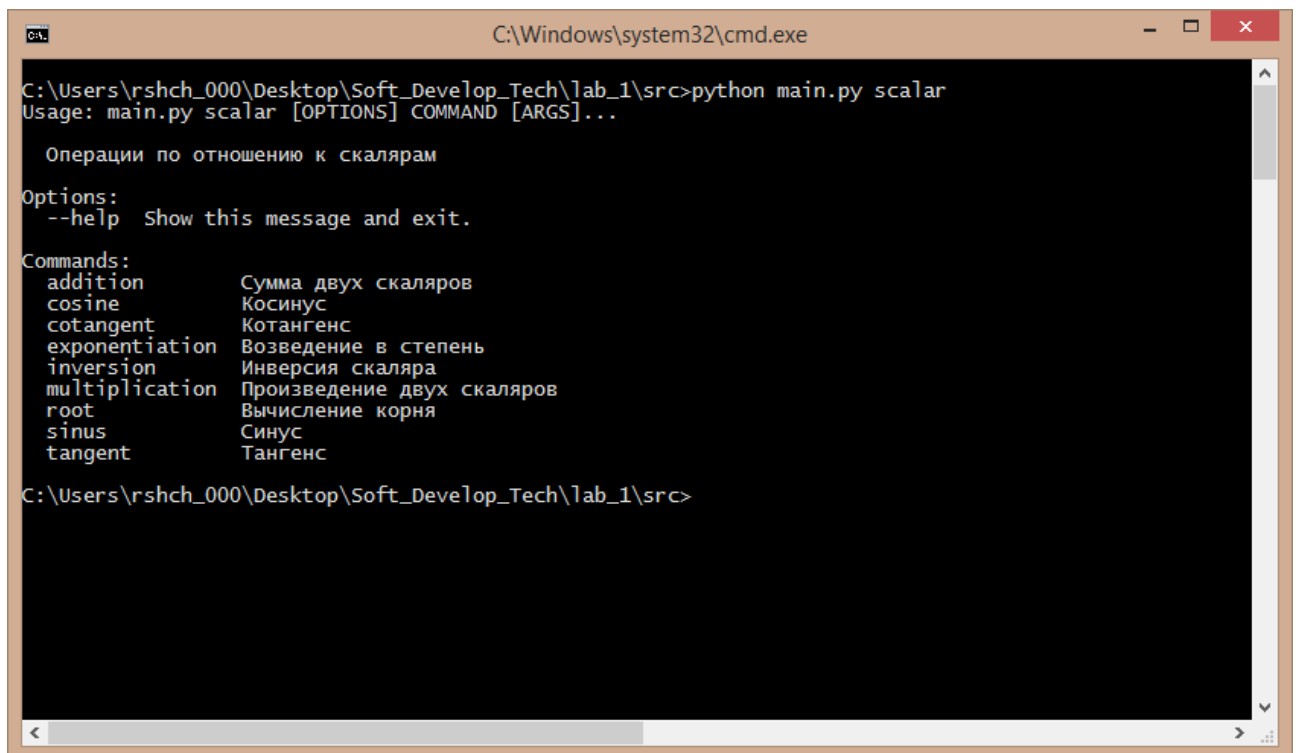


```
C:\Windows\system32\cmd.exe

C:\Users\rshch_000\Desktop\Soft_Develop_Tech\lab_1\src>python main.py vector codirect-vectors
Введите первый вектор. Пример: 1, 2, 3: 4, 16
Введите второй вектор. Пример: 1, 2, 3: 2, 8
Сонаправлены

C:\Users\rshch_000\Desktop\Soft_Develop_Tech\lab_1\src>
```

Рис. 8. Проверка сонаправленности векторов



```
C:\Windows\system32\cmd.exe

C:\Users\rshch_000\Desktop\Soft_Develop_Tech\lab_1\src>python main.py scalar
Usage: main.py scalar [OPTIONS] COMMAND [ARGS]...

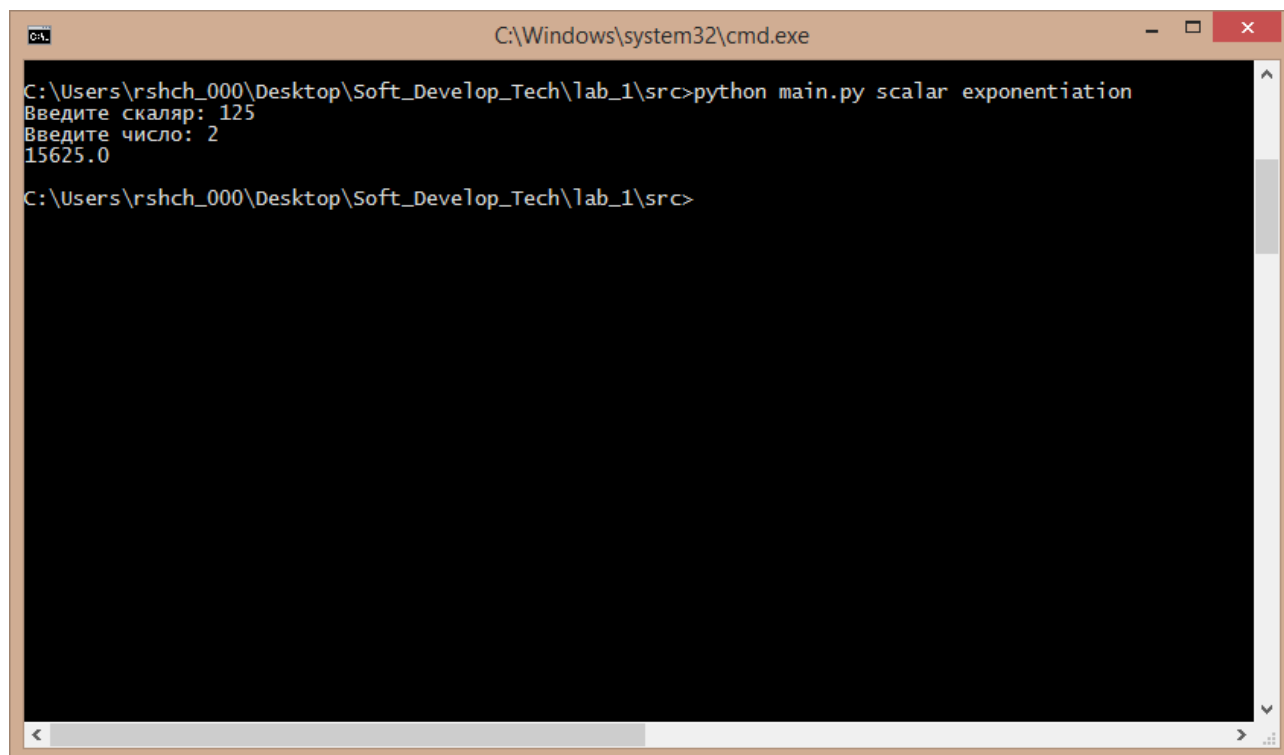
  Операции по отношению к скалярам

Options:
  --help  Show this message and exit.

Commands:
  addition      Сумма двух скаляров
  cosine        Косинус
  cotangent     Котангенс
  exponentiation Возведение в степень
  inversion      Инверсия скаляра
  multiplication Произведение двух скаляров
  root          Вычисление корня
  sinus         Синус
  tangent       Тангенс

C:\Users\rshch_000\Desktop\Soft_Develop_Tech\lab_1\src>
```

Рис. 9. Подкоманды для скаляра



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background with white text. The command prompt shows the following sequence of text:
C:\Users\rshch_000\Desktop\Soft_Develop_Tech\lab_1\src>python main.py scalar exponentiation
Введите скаляр: 125
Введите число: 2
15625.0
C:\Users\rshch_000\Desktop\Soft_Develop_Tech\lab_1\src>

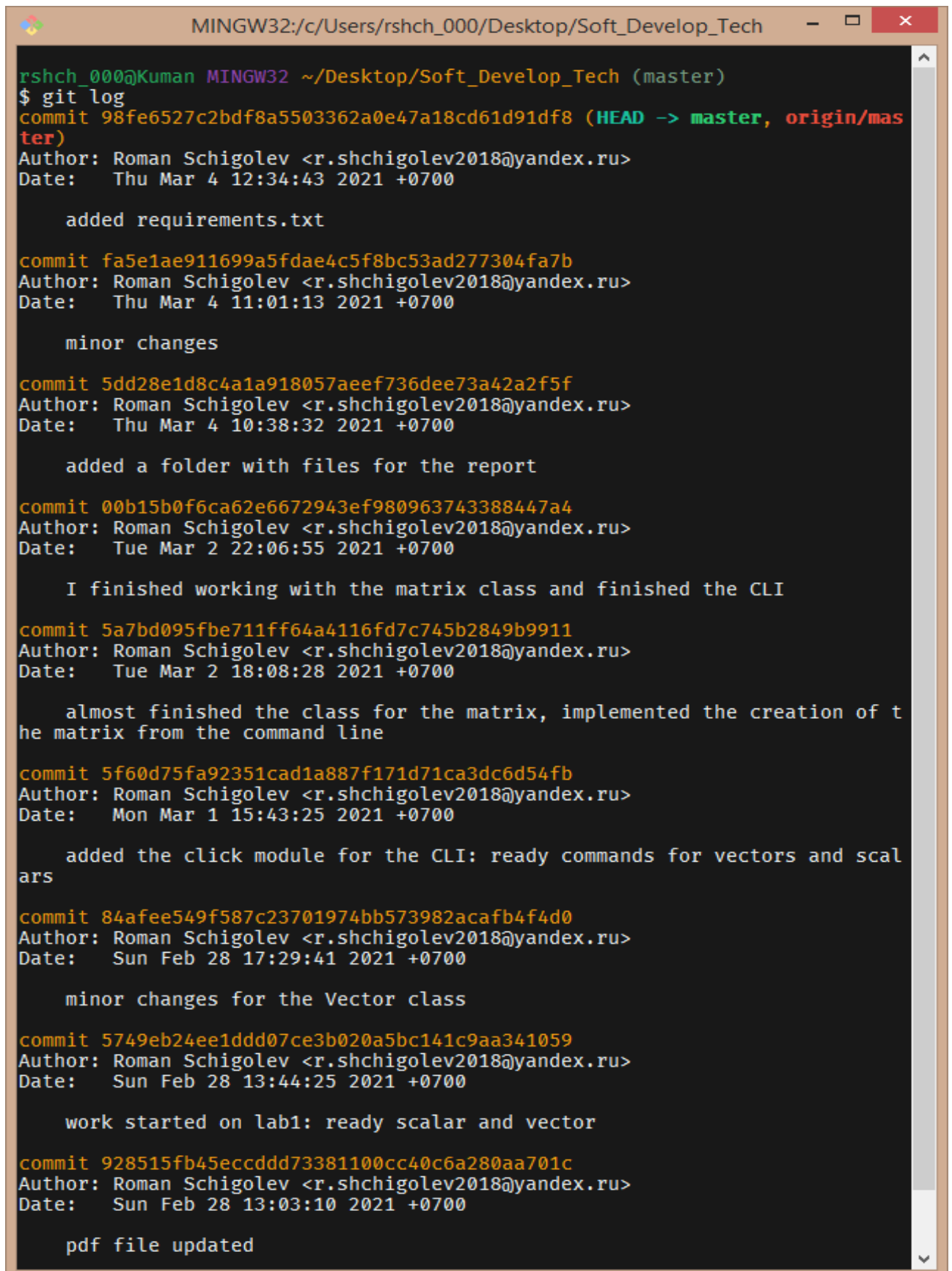
Рис. 10. Возведение в степень

Выводы по работе

В ходе выполнения лабораторной работы мы реализовали программу с команднострочным интерфейсом для выполнения различных математических преобразований над скалярными, векторными и тензорными величинами - скалярами, векторами и матрицами в объектно-ориентированном подходе с помощью языка программирования Python, модуля для создания команднострочного интерфейса Click и библиотеки NumPy.

Приложение

Вывод git log



```
MINGW32:/c/Users/rshch_000/Desktop/Soft_Develop_Tech
rshch_000@Kuman MINGW32 ~/Desktop/Soft_Develop_Tech (master)
$ git log
commit 98fe6527c2bdf8a5503362a0e47a18cd61d91df8 (HEAD -> master, origin/master)
Author: Roman Schigolev <r.shchigolev2018@yandex.ru>
Date: Thu Mar 4 12:34:43 2021 +0700

    added requirements.txt

commit fa5e1ae911699a5fdae4c5f8bc53ad277304fa7b
Author: Roman Schigolev <r.shchigolev2018@yandex.ru>
Date: Thu Mar 4 11:01:13 2021 +0700

    minor changes

commit 5dd28e1d8c4a1a918057aeef736dee73a42a2f5f
Author: Roman Schigolev <r.shchigolev2018@yandex.ru>
Date: Thu Mar 4 10:38:32 2021 +0700

    added a folder with files for the report

commit 00b15b0f6ca62e6672943ef980963743388447a4
Author: Roman Schigolev <r.shchigolev2018@yandex.ru>
Date: Tue Mar 2 22:06:55 2021 +0700

    I finished working with the matrix class and finished the CLI

commit 5a7bd095fbe711ff64a4116fd7c745b2849b9911
Author: Roman Schigolev <r.shchigolev2018@yandex.ru>
Date: Tue Mar 2 18:08:28 2021 +0700

    almost finished the class for the matrix, implemented the creation of the matrix from the command line

commit 5f60d75fa92351cad1a887f171d71ca3dc6d54fb
Author: Roman Schigolev <r.shchigolev2018@yandex.ru>
Date: Mon Mar 1 15:43:25 2021 +0700

    added the click module for the CLI: ready commands for vectors and scalars

commit 84afee549f587c23701974bb573982acafb4f4d0
Author: Roman Schigolev <r.shchigolev2018@yandex.ru>
Date: Sun Feb 28 17:29:41 2021 +0700

    minor changes for the Vector class

commit 5749eb24ee1ddd07ce3b020a5bc141c9aa341059
Author: Roman Schigolev <r.shchigolev2018@yandex.ru>
Date: Sun Feb 28 13:44:25 2021 +0700

    work started on lab1: ready scalar and vector

commit 928515fb45eccddd73381100cc40c6a280aa701c
Author: Roman Schigolev <r.shchigolev2018@yandex.ru>
Date: Sun Feb 28 13:03:10 2021 +0700

    pdf file updated
```

Рис. 11. Git log

UML-диаграмма

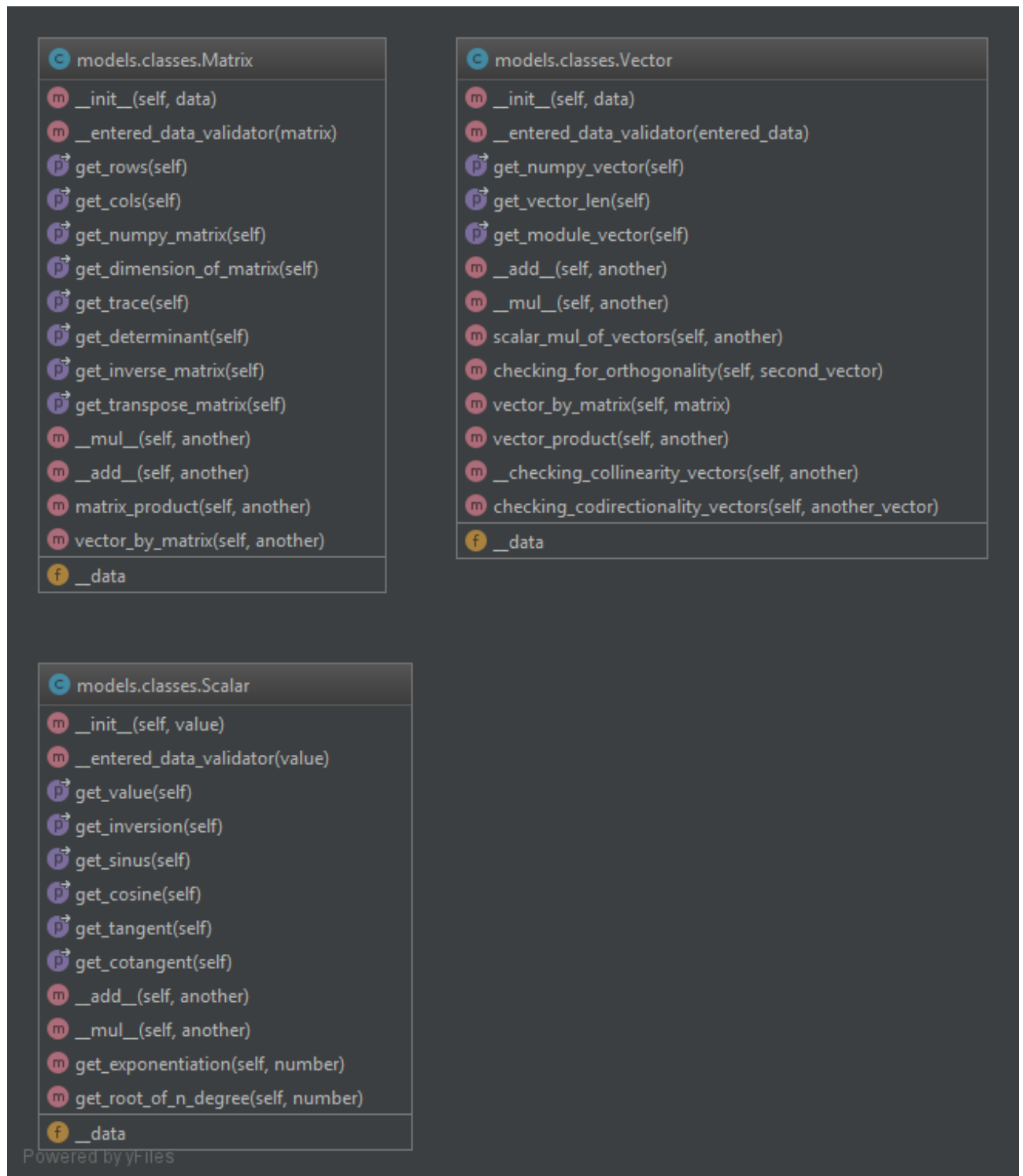


Рис. 12. UML-диаграмма

Листинг main.py

```
1  #!/usr/bin/env python
2
3  from models.classes import Vector, Scalar, Matrix
4  from utilities.parsing import matrix_parsing
5
```

```

6  import click
7
8
9  @click.group()
10 def cli():
11     '''Лаба 1'''
12     pass
13
14
15 @cli.group()
16 def scalar():
17     '''Операции по отношению к скалярам'''
18     pass
19
20
21 @scalar.command()
22 @click.option('--scalar1', prompt='Введите первый скаляр')
23 @click.option('--scalar2', prompt='Введите второй скаляр')
24 def addition(scalar1, scalar2):
25     '''Сумма двух скаляров'''
26     try:
27         first_scalar, second_scalar = Scalar(scalar1),
28         ↪ Scalar(scalar2)
29         result = first_scalar + second_scalar
30         click.echo(result)
31     except TypeError:
32         click.echo('Ошибка')
33
34 @scalar.command()
35 @click.option('--scal', prompt='Введите скаляр')
36 def inversion(scal):
37     '''Инверсия скаляра'''
38     try:
39         first_scalar = Scalar(scal)
40         result = first_scalar.get_inversion
41         click.echo(result)
42     except TypeError:
43         click.echo('Ошибка')
44
45
46 @scalar.command()
47 @click.option('--scalar1', prompt='Введите первый скаляр')
48 @click.option('--scalar2', prompt='Введите второй скаляр')

```

```

49 def multiplication(scalar1, scalar2):
50     '''Произведение двух скаляров'''
51     try:
52         first_scalar, second_scalar = Scalar(scalar1),
53         ↪ Scalar(scalar2)
54         result = first_scalar * second_scalar
55         click.echo(result)
56     except TypeError:
57         click.echo('Ошибка')
58
59 @scalar.command()
60 @click.option('--scal', prompt='Введите скаляр')
61 @click.option('--number', prompt='Введите число')
62 def exponentiation(scal, number):
63     '''Возведение в степень'''
64     try:
65         first_scalar, exponentiation_to_number = Scalar(scal),
66         ↪ number
67         result =
68         ↪ first_scalar.get_exponentiation(exponentiation_to_number)
69         click.echo(result)
70     except (TypeError, ValueError):
71         click.echo('Ошибка')
72
73 @scalar.command()
74 @click.option('--scal', prompt='Введите скаляр')
75 @click.option('--number', prompt='Введите число')
76 def root(scal, number):
77     '''Вычисление корня'''
78     try:
79         first_scalar, degree_root = Scalar(scal), number
80         result = first_scalar.get_root_of_n_degree(degree_root)
81         click.echo(result)
82     except (TypeError, ValueError):
83         click.echo('Ошибка')
84
85 @scalar.command()
86 @click.option('--scal', prompt='Введите скаляр')
87 def sinus(scal):
88     '''Синус'''
89     try:

```



```

90         first_scalar = Scalar(scal)
91         result = first_scalar.get_sinus
92         click.echo(result)
93     except TypeError:
94         click.echo('Ошибка')
95
96
97 @scalar.command()
98 @click.option('--scal', prompt='Введите скаляр')
99 def cosine(scal):
100     '''Косинус'''
101     try:
102         first_scalar = Scalar(scal)
103         result = first_scalar.get_cosine
104         click.echo(result)
105     except TypeError:
106         click.echo('Ошибка')
107
108
109 @scalar.command()
110 @click.option('--scal', prompt='Введите скаляр')
111 def tangent(scal):
112     '''Тангенс'''
113     try:
114         first_scalar = Scalar(scal)
115         result = first_scalar.get_tangent
116         click.echo(result)
117     except TypeError:
118         click.echo('Ошибка')
119
120
121 @scalar.command()
122 @click.option('--scal', prompt='Введите скаляр')
123 def cotangent(scal):
124     '''Котангенс'''
125     try:
126         first_scalar = Scalar(scal)
127         result = first_scalar.get_cotangent
128         click.echo(result)
129     except (TypeError, ZeroDivisionError):
130         click.echo('Ошибка')
131
132
133 @cli.group()

```

```

134 def vector():
135     '''Операции по отношению к векторам'''
136     pass
137
138
139 @vector.command()
140 @click.option('--vec', prompt='Введите вектор. Пример: 1, 2, 3')
141 @click.option('--scal', prompt='Введите скаляр')
142 def vec_by_sclar(vec, scal):
143     '''Умножение вектора на скаляр'''
144     try:
145         first_vector, second_sclar = Vector(vec), Sclar(scal)
146         result = first_vector * second_sclar
147         click.echo(result)
148     except TypeError:
149         click.echo('Ошибка')
150
151
152 @vector.command()
153 @click.option('--vector1', prompt='Введите первый вектор. Пример:
    ↪ 1, 2, 3')
154 @click.option('--vector2', prompt='Введите второй вектор. Пример:
    ↪ 1, 2, 3')
155 def el_by_el_add(vector1, vector2):
156     '''Поэлементное сложение'''
157     try:
158         first_vector, second_vector = Vector(vector1),
    ↪ Vector(vector2)
159         result = first_vector + second_vector
160         click.echo(result)
161     except TypeError:
162         click.echo('Ошибка')
163
164
165 @vector.command()
166 @click.option('--vector1', prompt='Введите первый вектор. Пример:
    ↪ 1, 2, 3')
167 @click.option('--vector2', prompt='Введите второй вектор. Пример:
    ↪ 1, 2, 3')
168 def el_by_el_mul(vector1, vector2):
169     '''Поэлементное умножение'''
170     try:
171         first_vector, second_vector = Vector(vector1),
    ↪ Vector(vector2)

```

```

172         result = first_vector * second_vector
173         click.echo(result)
174     except TypeError:
175         click.echo('Ошибка')
176     except ValueError:
177         click.echo('Разная длина векторов')
178
179
180 @vector.command()
181 @click.option('--vec', prompt='Введите вектор. Пример: 1, 2, 3')
182 @click.option('--rows', prompt='Введите количество строк для
    ↪ матрицы', type=int)
183 @click.option('--cols', prompt='Введите количество столбцов для
    ↪ матрицы', type=int)
184 def vec_by_matrix(vec, rows, cols):
185     '''Умножение вектора на матрицу'''
186     number_of_rows = rows
187     number_of_cols = cols
188     try:
189         vec = Vector(vec)
190         mtrx = Matrix(matrix_parsing(number_of_rows,
    ↪ number_of_cols))
191         result = vec.vector_by_matrix(mtrx)
192         click.echo(result)
193     except ValueError:
194         click.echo('Ошибка')
195     except TypeError:
196         click.echo('Возникла ошибка')
197
198
199 @vector.command()
200 @click.option('--vector1', prompt='Введите первый вектор. Пример:
    ↪ 1, 2, 3')
201 @click.option('--vector2', prompt='Введите второй вектор. Пример:
    ↪ 1, 2, 3')
202 def scalar_product(vector1, vector2):
203     '''Скалярное произведение'''
204     try:
205         first_vector, second_vector = Vector(vector1),
    ↪ Vector(vector2)
206         result = first_vector.scalar_mul_of_vectors(second_vector)
207         click.echo(result)
208     except TypeError:
209         click.echo('Ошибка')

```

```

210     except ValueError:
211         click.echo('Разная длина векторов')
212
213
214 @vector.command()
215 @click.option('--vector1', prompt='Введите первый вектор. Пример:
    ↪ 1, 2, 3')
216 @click.option('--vector2', prompt='Введите второй вектор. Пример:
    ↪ 1, 2, 3')
217 def vector_product(vector1, vector2):
218     '''Векторное произведение'''
219     try:
220         first_vector, second_vector = Vector(vector1),
    ↪ Vector(vector2)
221         result = first_vector.vector_product(second_vector)
222         click.echo(result)
223     except TypeError:
224         click.echo('Ошибка')
225     except ValueError:
226         click.echo('Вектор не трехмерный')
227
228
229 @vector.command()
230 @click.option('--vector1', prompt='Введите вектор. Пример: 1, 2,
    ↪ 3')
231 def module_vector(vector1):
232     '''Вычисление длины (модуля) вектора'''
233     try:
234         first_vector = Vector(vector1)
235         result = first_vector.get_module_vector
236         click.echo(result)
237     except TypeError:
238         click.echo('Ошибка')
239
240
241 @vector.command()
242 @click.option('--vector1', prompt='Введите первый вектор. Пример:
    ↪ 1, 2, 3')
243 @click.option('--vector2', prompt='Введите второй вектор. Пример:
    ↪ 1, 2, 3')
244 def codirect_vectors(vector1, vector2):
245     '''Проверка сонаправленности векторов'''
246     try:

```

```

247         first_vector, second_vector = Vector(vector1),
        ↪ Vector(vector2)
248     result =
        ↪ first_vector.checking_codirectionality_vectors(second_vector)
249     click.echo('Сонаправлены' if result else 'Несонаправлены')
250 except ValueError:
251     click.echo('Ошибка. Векторы не должны быть нулевыми и
        ↪ должны иметь одинаковую длину')
252 except TypeError:
253     click.echo('Ошибка')
254
255
256 @vector.command()
257 @click.option('--vector1', prompt='Введите первый вектор. Пример:
        ↪ 1, 2, 3')
258 @click.option('--vector2', prompt='Введите второй вектор. Пример:
        ↪ 1, 2, 3')
259 def orthog_vectors(vector1, vector2):
260     '''Проверка векторов на ортогональность'''
261     try:
262         first_vector, second_vector = Vector(vector1),
        ↪ Vector(vector2)
263         result =
        ↪ first_vector.checking_for_orthogonality(second_vector)
264         click.echo('Ортогональны' if result else 'Неортогональны')
265     except ValueError:
266         click.echo('Разная длина векторов')
267     except TypeError:
268         click.echo('Ошибка')
269
270
271 @cli.group()
272 def matrix():
273     '''Операции по отношению к матрицам'''
274
275
276 @matrix.command()
277 @click.option('--scal', prompt='Введите скаляр')
278 @click.option('--rows', prompt='Введите количество строк для
        ↪ матрицы', type=int)
279 @click.option('--cols', prompt='Введите количество столбцов для
        ↪ матрицы', type=int)
280 def matrix_by_scalar(scal, rows, cols):
281     '''Умножение матрицы на скаляр'''

```

```

282     number_of_rows = rows
283     number_of_cols = cols
284     try:
285         scal = Scalar(scal)
286         mtrx = Matrix(matrix_parsing(number_of_rows,
287                                     ↪ number_of_cols))
288         result = mtrx * scal
289         click.echo(result)
290     except ValueError:
291         click.echo('Ошибка')
292     except TypeError:
293         click.echo('Возникла ошибка. Скаляр - число')
294
295 @matrix.command()
296 @click.option('--rows1', prompt='Введите количество строк для
297 ↪ первой матрицы', type=int)
298 @click.option('--cols1', prompt='Введите количество столбцов для
299 ↪ первой матрицы', type=int)
300 @click.option('--rows2', prompt='Введите количество строк для
301 ↪ второй матрицы', type=int)
302 @click.option('--cols2', prompt='Введите количество столбцов для
303 ↪ второй матрицы', type=int)
304 def el_by_el_add(rows1, cols1, rows2, cols2):
305     '''Поэлементное сложение'''
306     number_of_rows_for_first_matrix,
307     ↪ number_of_cols_for_first_matrix = rows1, cols1
308     number_of_rows_for_second_matrix,
309     ↪ number_of_cols_for_second_matrix = rows2, cols2
310     try:
311         first_matrix =
312         ↪ Matrix(matrix_parsing(number_of_rows_for_first_matrix,
313                                 ↪ number_of_cols_for_first_matrix))
314         second_matrix =
315         ↪ Matrix(matrix_parsing(number_of_rows_for_second_matrix,
316                                 ↪ number_of_cols_for_second_matrix))
317         result = first_matrix + second_matrix
318         click.echo(result)
319     except ValueError:
320         click.echo('Размеры матриц должны совпадать')
321     except TypeError:
322         click.echo('Это не матрица')
323
324
325

```

```

315 @matrix.command()
316 @click.option('--rows1', prompt='Введите количество строк для
    ↪ первой матрицы', type=int)
317 @click.option('--cols1', prompt='Введите количество столбцов для
    ↪ первой матрицы', type=int)
318 @click.option('--rows2', prompt='Введите количество строк для
    ↪ второй матрицы', type=int)
319 @click.option('--cols2', prompt='Введите количество столбцов для
    ↪ второй матрицы', type=int)
320 def el_by_el_mul(rows1, cols1, rows2, cols2):
321     '''Поэлементное произведение'''
322     number_of_rows_for_first_matrix,
    ↪ number_of_cols_for_first_matrix = rows1, cols1
323     number_of_rows_for_second_matrix,
    ↪ number_of_cols_for_second_matrix = rows2, cols2
324     try:
325         first_matrix =
    ↪ Matrix(matrix_parsing(number_of_rows_for_first_matrix,
    ↪ number_of_cols_for_first_matrix))
326         second_matrix =
    ↪ Matrix(matrix_parsing(number_of_rows_for_second_matrix,
    ↪ number_of_cols_for_second_matrix))
327         result = first_matrix * second_matrix
328         click.echo(result)
329     except ValueError:
330         click.echo('Размеры матриц должны совпадать')
331     except TypeError:
332         click.echo('Это не матрица')
333
334
335 @matrix.command()
336 @click.option('--vec', prompt='Введите вектор. Пример: 1, 2, 3')
337 @click.option('--rows', prompt='Введите количество строк для
    ↪ матрицы', type=int)
338 @click.option('--cols', prompt='Введите количество столбцов для
    ↪ матрицы', type=int)
339 def vec_by_matrix(vec, rows, cols):
340     '''Умножение вектора на матрицу'''
341     number_of_rows = rows
342     number_of_cols = cols
343     try:
344         vec = Vector(vec)
345         mtrx = Matrix(matrix_parsing(number_of_rows,
    ↪ number_of_cols))

```

```

346         result = vec.vector_by_matrix(mtrx)
347         click.echo(result)
348     except ValueError:
349         click.echo('Ошибка')
350     except TypeError:
351         click.echo('Возникла ошибка')
352
353
354 @matrix.command()
355 @click.option('--rows1', prompt='Введите количество строк для
    ↪ первой матрицы', type=int)
356 @click.option('--cols1', prompt='Введите количество столбцов для
    ↪ первой матрицы', type=int)
357 @click.option('--rows2', prompt='Введите количество строк для
    ↪ второй матрицы', type=int)
358 @click.option('--cols2', prompt='Введите количество столбцов для
    ↪ второй матрицы', type=int)
359 def matrix_product(rows1, cols1, rows2, cols2):
360     '''Матричное произведение'''
361     number_of_rows_for_first_matrix,
    ↪ number_of_cols_for_first_matrix = rows1, cols1
362     number_of_rows_for_second_matrix,
    ↪ number_of_cols_for_second_matrix = rows2, cols2
363     try:
364         first_matrix =
    ↪ Matrix(matrix_parsing(number_of_rows_for_first_matrix,
    ↪ number_of_cols_for_first_matrix))
365         second_matrix =
    ↪ Matrix(matrix_parsing(number_of_rows_for_second_matrix,
    ↪ number_of_cols_for_second_matrix))
366         result = first_matrix.matrix_product(second_matrix)
367         click.echo(result)
368     except ValueError:
369         click.echo('Количество столбцов первой матрицы не равно
    ↪ количеству строк второй матрицы')
370     except TypeError:
371         click.echo('Это не матрица')
372
373
374 @matrix.command()
375 @click.option('--rows', prompt='Введите количество строк для
    ↪ матрицы', type=int)
376 @click.option('--cols', prompt='Введите количество столбцов для
    ↪ матрицы', type=int)

```



```

377 def trace(rows, cols):
378     '''Вычисление следа'''
379     number_of_rows, number_of_cols = rows, cols
380     try:
381         mtrx = Matrix(matrix_parsing(number_of_rows,
382                                     ↪ number_of_cols))
383         result = mtrx.get_trace
384         click.echo(result)
385     except ValueError:
386         click.echo('Количество столбцов первой матрицы не равно
387                     ↪ количеству строк второй матрицы')
388     except TypeError:
389         click.echo('Возникла ошибка')
390
391 @matrix.command()
392 @click.option('--rows', prompt='Введите количество строк для
393               ↪ матрицы', type=int)
394 @click.option('--cols', prompt='Введите количество столбцов для
395               ↪ матрицы', type=int)
396 def determinant(rows, cols):
397     '''Вычисление определителя'''
398     number_of_rows, number_of_cols = rows, cols
399     if number_of_rows != number_of_cols:
400         click.echo('Количество строк должно совпадать с
401                     ↪ количеством столбцов')
402         return
403     try:
404         mtrx = Matrix(matrix_parsing(number_of_rows,
405                                     ↪ number_of_cols))
406         result = mtrx.get_determinant
407         click.echo(result)
408     except ValueError:
409         click.echo('Ошибка')
410     except TypeError:
411         click.echo('Возникла ошибка')
412
413 @matrix.command()
414 @click.option('--rows', prompt='Введите количество строк для
415               ↪ матрицы', type=int)
416 @click.option('--cols', prompt='Введите количество столбцов для
417               ↪ матрицы', type=int)
418 def inverse_matrix(rows, cols):

```

```

413     '''Вычисление обратной матрицы'''
414     number_of_rows, number_of_cols = rows, cols
415     if number_of_rows != number_of_cols:
416         click.echo('Количество строк должно совпадать с
417             ↳ количеством столбцов')
418         return
419     try:
420         mtrx = Matrix(matrix_parsing(number_of_rows,
421             ↳ number_of_cols))
422         result = mtrx.get_inverse_matrix
423         click.echo(result)
424     except ValueError:
425         click.echo('Ошибка')
426     except TypeError:
427         click.echo('Возникла ошибка')
428
429 @matrix.command()
430 @click.option('--rows', prompt='Введите количество строк для
431     ↳ матрицы', type=int)
432 @click.option('--cols', prompt='Введите количество столбцов для
433     ↳ матрицы', type=int)
434 def transpose_matrix(rows, cols):
435     '''Транспонирование'''
436     number_of_rows, number_of_cols = rows, cols
437     try:
438         mtrx = Matrix(matrix_parsing(number_of_rows,
439             ↳ number_of_cols))
440         result = mtrx.get_transpose_matrix
441         click.echo(result)
442     except ValueError:
443         click.echo('Ошибка')
444     except TypeError:
445         click.echo('Возникла ошибка')
446
447 if __name__ == '__main__':
448     cli()

```

Листинг classes.py

```

1  from math import (sin, cos, tan)
2  import re
3

```

```

4 import numpy as np
5
6
7 class Scalar:
8     def __init__(self, value):
9         if Scalar.__entered_data_validator(value):
10             self.__data = float(value)
11         else:
12             raise TypeError('Это не число')
13
14     @staticmethod
15     def __entered_data_validator(value):
16         regexp_query =
17             → re.compile(r'[+-]?((([1-9][0-9]*)|(0))([.],[0-9]+)?')
18         return regexp_query.fullmatch(value)
19
20     @property
21     def get_value(self):
22         return self.__data
23
24     @property
25     def get_inversion(self):
26         return -self.__data
27
28     @property
29     def get_sinus(self):
30         return sin(self.__data)
31
32     @property
33     def get_cosine(self):
34         return cos(self.__data)
35
36     @property
37     def get_tangent(self):
38         return tan(self.__data)
39
40     @property
41     def get_cotangent(self):
42         return self.get_cosine / self.get_sinus
43
44     def __add__(self, another):
45         if isinstance(another, Scalar):
46             return self.__data + another.get_value
47         return self.__data + another

```

```

47
48     def __mul__(self, another):
49         if isinstance(another, Scalar):
50             return self.__data * another.get_value
51         return self.__data * another
52
53     def get_exponentiation(self, number):
54         return pow(self.get_value, float(number))
55
56     def get_root_of_n_degree(self, number):
57         return pow(self.get_value, 1 / float(number))
58
59
60 class Vector:
61     def __init__(self, data):
62         if Vector.__entered_data_validator(data) and len(data) !=
        ↪ 0:
63             modified_data = [float(i) for i in data.split(', ')]
64             self.__data = np.array(modified_data)
65         elif len(data) == 0:
66             self.__data = np.zeros(2)
67         else:
68             raise TypeError('Введенные данные не являются числами')
69
70     @staticmethod
71     def __entered_data_validator(entered_data):
72         modified_data = [i for i in entered_data.split(', ')]
73         regexp_query =
        ↪ re.compile(r'[+-]?((([1-9][0-9]*)|(0))([.][0-9]+)?')
74         counter_correct_items = 0
75         for item in modified_data:
76             if regexp_query.fullmatch(item):
77                 counter_correct_items += 1
78         return len(modified_data) == counter_correct_items
79
80     @property
81     def get_numpy_vector(self):
82         return self.__data
83
84     @property
85     def get_vector_len(self):
86         return len(self.__data)
87
88     @property

```

```

89     def get_module_vector(self):
90         return np.linalg.norm(self.__data)
91
92     def __add__(self, another):
93         if not isinstance(another, Vector):
94             raise TypeError('Это не вектор')
95         if self.get_vector_len != another.get_vector_len:
96             raise ValueError('Разные длины векторов')
97         return self.__data + another.get_numpy_vector
98
99     def __mul__(self, another):
100         if isinstance(another, (int, float)):
101             return self.__data * another
102         if isinstance(another, Scalar):
103             return self.__data * another.get_value
104         if isinstance(another, Vector):
105             if self.get_vector_len != another.get_vector_len:
106                 raise ValueError('Разные длины векторов')
107             return self.__data * another.get_numpy_vector
108
109     def scalar_mul_of_vectors(self, another):
110         if not isinstance(another, Vector):
111             raise TypeError('Это не вектор')
112         if self.get_vector_len != another.get_vector_len:
113             raise ValueError('Разные длины векторов')
114         return np.dot(self.__data, another.get_numpy_vector)
115
116     def checking_for_orthogonality(self, second_vector):
117         if not isinstance(second_vector, Vector):
118             raise TypeError('Это не вектор')
119         return self.scalar_mul_of_vectors(second_vector) == 0
120
121     def vector_by_matrix(self, matrix):
122         if not isinstance(matrix, Matrix):
123             raise TypeError('Это не матрица')
124         if self.get_vector_len != matrix.get_rows:
125             raise ValueError(
126                 'Число столбцов в матрице должно совпадать с
127                 ↪ числом строк в векторе-столбце')
128         return np.dot(self.get_numpy_vector,
129             ↪ matrix.get_numpy_matrix)
130
131     def vector_product(self, another):
132         if not isinstance(another, Vector):

```

```

131         raise TypeError('Это не вектор')
132     if self.get_vector_len < 3 or another.get_vector_len < 3:
133         raise ValueError('Вектор не трехмерный')
134     return np.cross(self.__data, another.get_numpy_vector)
135
136     def __checking_collinearity_vectors(self, another):
137         if not isinstance(another, Vector):
138             raise TypeError('Это не вектор')
139         if not np.any(self.get_numpy_vector) or not
140             ↪ np.any(another.get_numpy_vector):
141             raise ValueError(
142                 'Один из векторов равен нулю, поэтому вопрос о
143                 ↪ коллинеарности векторов некорректен!')
144         if self.get_vector_len != another.get_vector_len:
145             raise ValueError('Разные длины векторов')
146
147         first_vector = self.__data
148         second_vector = another.get_numpy_vector
149         set_coordinate_relations = []
150
151         if len(first_vector[first_vector > 0]) ==
152             ↪ self.get_vector_len and \
153             len(second_vector[second_vector > 0]) ==
154             ↪ len(second_vector) or \
155             np.count_nonzero(first_vector) !=
156             ↪ self.get_vector_len and \
157             len(second_vector[second_vector > 0]) ==
158             ↪ len(second_vector):
159
160             for value_first_vector, value_second_vector in
161                 ↪ zip(first_vector, second_vector):
162                 set_coordinate_relations.append(
163                     value_first_vector / value_second_vector)
164
165             return len(set(set_coordinate_relations)) == 1
166
167         if len(second_vector[second_vector > 0]) !=
168             ↪ len(second_vector):
169             index_nonzero_element = np.where(first_vector !=
170                 ↪ 0)[0][0]
171             scalar = second_vector[index_nonzero_element] /
172                 ↪ first_vector[index_nonzero_element]
173             intermediate_vector = first_vector * scalar

```

```

165         return np.array_equal(intermediate_vector,
166                                ↪ second_vector)
167
168     def checking_codirectionality_vectors(self, another_vector):
169         if self.__checking_collinearity_vectors(another_vector):
170             return self.scalar_mul_of_vectors(another_vector) > 0
171         return False
172
173     class Matrix:
174         def __init__(self, data):
175             if Matrix.__entered_data_validator(data):
176                 self.__data = np.matrix(data)
177             else:
178                 raise TypeError('Неправильная матрица')
179
180         @staticmethod
181         def __entered_data_validator(matrix):
182             if len(matrix) != 0 and len(matrix[0]) != 0:
183                 for row in matrix:
184                     if len(row) != len(matrix[0]):
185                         return False
186                     for element in row:
187                         if not isinstance(element, (int, float)):
188                             return False
189             return True
190         return False
191
192         @property
193         def get_rows(self):
194             return self.__data.shape[0]
195
196         @property
197         def get_cols(self):
198             return self.__data.shape[1]
199
200         @property
201         def get_numpy_matrix(self):
202             return self.__data.A
203
204         @property
205         def get_dimension_of_matrix(self):
206             return self.get_numpy_matrix.shape
207

```

```

208 @property
209 def get_trace(self):
210     return np.trace(self.get_numpy_matrix)
211
212 @property
213 def get_determinant(self):
214     return np.linalg.det(self.get_numpy_matrix)
215
216 @property
217 def get_inverse_matrix(self):
218     return np.linalg.inv(self.get_numpy_matrix)
219
220 @property
221 def get_transpose_matrix(self):
222     return self.get_numpy_matrix.T
223
224 def __mul__(self, another):
225     if isinstance(another, Scalar):
226         return self.get_numpy_matrix.dot(another.get_value)
227     if isinstance(another, Matrix):
228         if self.get_dimension_of_matrix !=
229             ↪ another.get_dimension_of_matrix:
230             raise ValueError('Размеры матриц должны
231                 ↪ совпадать')
232         return np.multiply(self.get_numpy_matrix,
233             ↪ another.get_numpy_matrix)
234     else:
235         return self.get_numpy_matrix * another
236
237 def __add__(self, another):
238     if not isinstance(another, Matrix):
239         raise TypeError('Это не матрица')
240     if self.get_dimension_of_matrix !=
241         ↪ another.get_dimension_of_matrix:
242         raise ValueError('Размеры матриц должны совпадать')
243     return self.get_numpy_matrix + another.get_numpy_matrix
244
245 def matrix_product(self, another):
246     if not isinstance(another, Matrix):
247         raise TypeError('Это не матрица')
248     if self.get_cols != another.get_rows:
249         raise ValueError('Количество столбцов первой матрицы
250             ↪ не равно количеству строк второй матрицы')

```



```

246         return np.dot(self.get_numpy_matrix,
    ↪     another.get_numpy_matrix)
247
248     def vector_by_matrix(self, another):
249         if not isinstance(another, Vector):
250             raise TypeError('Это не вектор')
251         if another.get_vector_len != self.get_rows:
252             raise ValueError('Разная длина')
253         return np.dot(another.get_numpy_vector,
    ↪     self.get_numpy_matrix)

```

Листинг parsing.py

```

1  def matrix_parsing(rows, cols):
2      matrix = []
3
4      if rows == 0 or cols == 0:
5          print('Ошибка')
6          return
7
8      try:
9          for row in range(1, rows + 1):
10             elements = input(f'Введите элементы для {row}-ой
    ↪ строки через пробел: ')
11             subarray = list(map(float, elements.split()))
12             if len(subarray) == cols:
13                 matrix.append(subarray)
14             else:
15                 raise ValueError('Превышение по количеству
    ↪ столбцов')
16         return matrix
17     except:
18         raise ValueError('Ошибка')

```