**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра информационных систем**

# ОТЧЕТ

**по практической работе №1**
**по дисциплине «Объектно-ориентированное программирование»**
**Тема: Консольное приложение для вычислений над квадратной матрицей, заданной вещественными числами.**

Студенты гр. 1361 _____ Горбунова Д.А.
Кравцов И.Ю.

Преподаватель _____ Егоров С.С.

Санкт-Петербург
2024

## ЗАДАНИЕ НА ПРАКТИЧЕСКУЮ РАБОТУ

Тема работы: консольное приложение согласно представленной на рисунке 1 диаграмме классов, предназначенное для заданных вычислений над квадратной матрицей, заданной на множестве вещественных чисел.

Исходные данные:

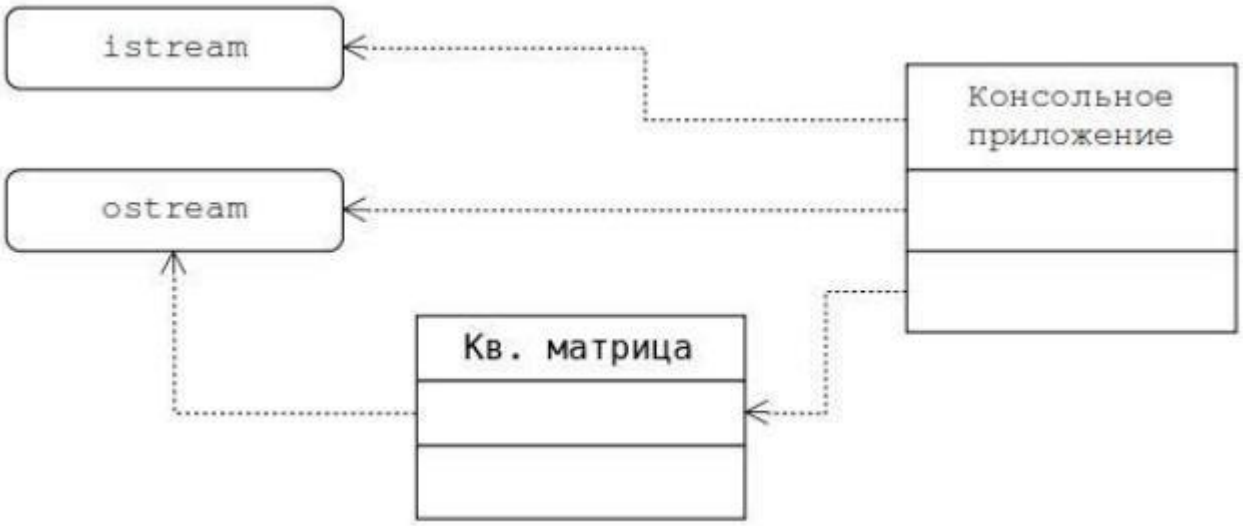


Рис.1 – Диаграмма классов

Описание работы:

Для выполнения практической работы необходимо специфицировать пользовательские классы "Консольное приложение " и "Квадратная матрица", т.е. задать атрибуты и методы указанных классов, а также распределить их по существующим областям видимости. Спецификация классов и реализация их методов должна обеспечивать реализацию отношений, указанных на диаграмме классов. В отчете представить аргументированное обоснование своего выбора.

# СОДЕРЖАНИЕ

# 1. СПЕЦИФИКАЦИЯ РАЗРАБОТАННЫХ КЛАССОВ

## 1.1. Класс `SquareMatrix` («Квадратная матрица»):

Атрибуты класса:

a) `int m_rows, m_cols`

Область видимости: private.

Описание: хранение размерности матрицы.

Стандартное значение: 3.

b) `vector<vector<number>> m_values`

Область видимости: private.

Описание: вектор, содержащий вектора-строки матрицы.

Стандартное значение: Единичная матрица.

Чтобы предотвратить несанкционированный доступ, атрибуты класса ограничены доступом и могут быть использованы только внутри модуля, где определен класс квадратной матрицы.

Методы класса:

a) `void generate_1()`

Область видимости: public.

Назначение: заполняет единичную матрицу.

b) `void print_screen()`

Область видимости: public.

Назначение: выводит текущую матрицу.

c) `SquareMatrix transpose()`

Область видимости: public.

Назначение: транспонирует текущую матрицу.

d) `number determinant()const`

Область видимости: public.

Назначение: вычисляет определитель текущей матрицы.

e) `int rows() const`

Область видимости: public.

Назначение: возвращает количество строк в текущей матрице.

f) `int cols() const`

Область видимости: public.

Назначение: возвращает количество столбцов в текущей матрице.

g) `int rank()`

Область видимости: public.

Назначение: возвращает ранг текущей матрицы.

h) `number determinantHelper (const std::vector<std::vector<number>>& matrix) const`

Область видимости: private.

Назначение: вычисляет определитель матрицы меньшего размера.

Метод `determinantHelper` имеет тип доступа private т.к. для его работы необходим доступ к приватным полям класса. Остальные методы класса являются общедоступными т.к. задействуются в других модулях и используют публичные поля для работы.

1.2. Класс `ConsoleApplication` («Консольное приложение»):

Атрибуты класса:

Не имеется.

Методы класса:

a) `int exec()`

Область видимости: public.

Назначение: запускает работу приложения.

b) `int menu()`

Область видимости: private.

Назначение: предоставляет выбор методов для пользователя.

Метод `menu` имеет закрытый тип доступа т. к. используется исключительно в модуле `Application`.

# 2. ДИАГРАММА КЛАССОВ

Диаграмма классов представлена на рисунке 2. На ней обозначены атрибуты классов и их методы. Знаками «+» обозначены методы, атрибуты имеющие общедоступный тип доступа; соответственно «-» – приватный. Также через двоеточие указаны типы возвращаемых значений, где они есть.

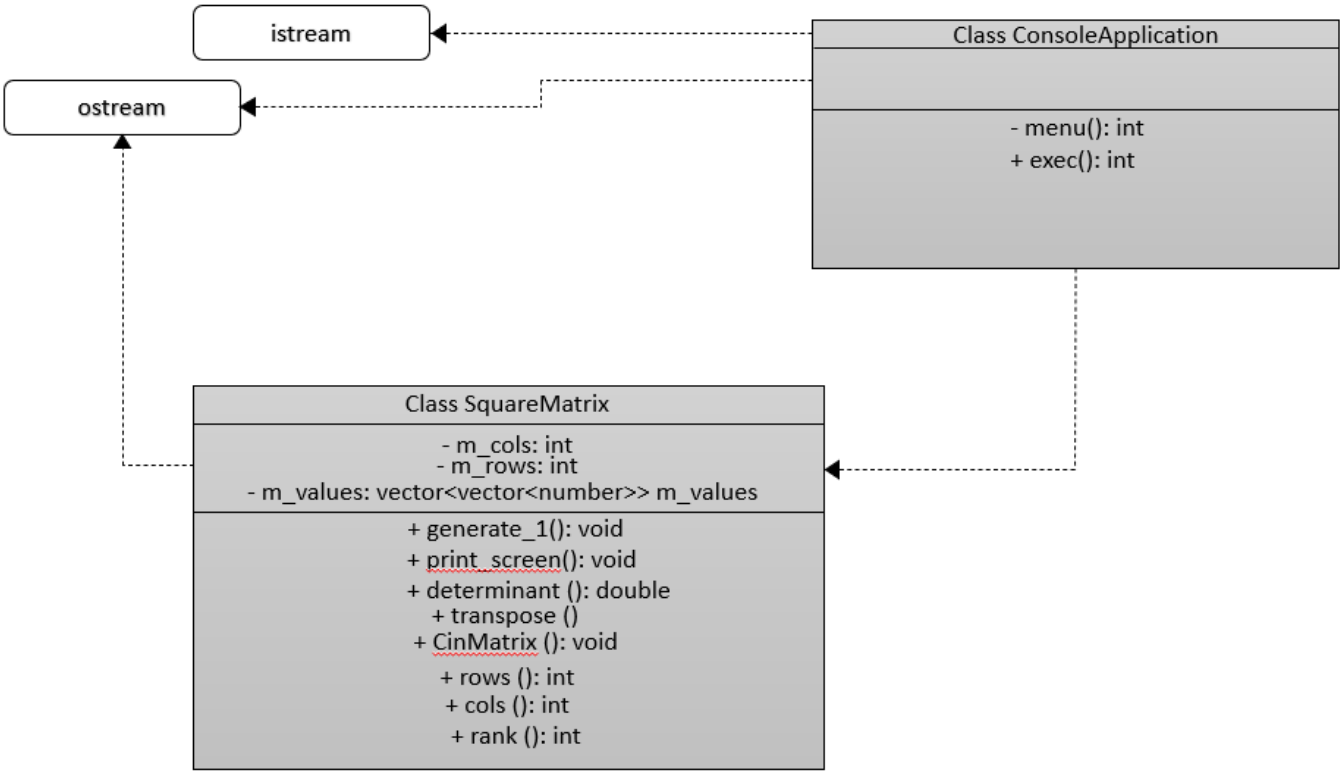


Рисунок 2 – Диаграмма классов

# 3. ОПИСАНИЕ КОНТРОЛЬНОГО ПРИМЕРА

Исходные данные для контрольного примера:

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$$

Чтобы найти транспонированную матрицу поменяем строки столбцы матрицы местами:

$$\mathbf{A}^{\mathbf{T}} = \begin{pmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{pmatrix}$$

Далее ищем ранг матрицы:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} => \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & -4 & -8 & -12 \\ 0 & -8 & -16 & -24 \\ 0 & -12 & -24 & -36 \end{pmatrix} => \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 \\ 0 & -8 & -16 & -24 \\ 0 & -12 & -24 & -36 \end{pmatrix}$$

$$=> \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} =>$$ т.к. ненулевых строк 2, то ранг матрицы = 2.

Теперь вычислим определитель матрицы:

$$\det \mathbf{A} = \begin{vmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{vmatrix} => \begin{vmatrix} 1 & 2 & 3 & 4 \\ 0 & -4 & -8 & -12 \\ 0 & -8 & -16 & -24 \\ 0 & -12 & -24 & -36 \end{vmatrix} => \begin{vmatrix} 1 & 2 & 3 & 4 \\ 0 & -4 & -8 & -12 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

= 1 * (-4) * 0 * 0 = 0.

# 4. РАБОТА ПРОГРАММЫ НА КОНТРОЛЬНЫХ ПРИМЕРАХ

При запуске приложения, программа выводит на экран доступные варианты работы с матрицей (рисунок 3).

```
 0 -- exit
1 -- input of matrix coefficients
2 -- calculate determinant
3 -- transpose matrix
4 -- calculate rank
5 -- print matrix
```

Рисунок 3 – Меню

Далее производим ввод пользовательской матрицы с клавиатуры.

```
> 1
 enter matrix dimensions
n=4
 enter coefficient matrices

matrix [0][0] =1

matrix [0][1] =2

matrix [0][2] =3

matrix [0][3] =4

matrix [1][0] =5

matrix [1][1] =6

matrix [1][2] =7

matrix [1][3] =8

matrix [2][0] =9

matrix [2][1] =10

matrix [2][2] =11

matrix [2][3] =12

matrix [3][0] =13

matrix [3][1] =14

matrix [3][2] =15

matrix [3][3] =16
```

После того мы определили матрицу, ее можно вывести (рисунок 4).

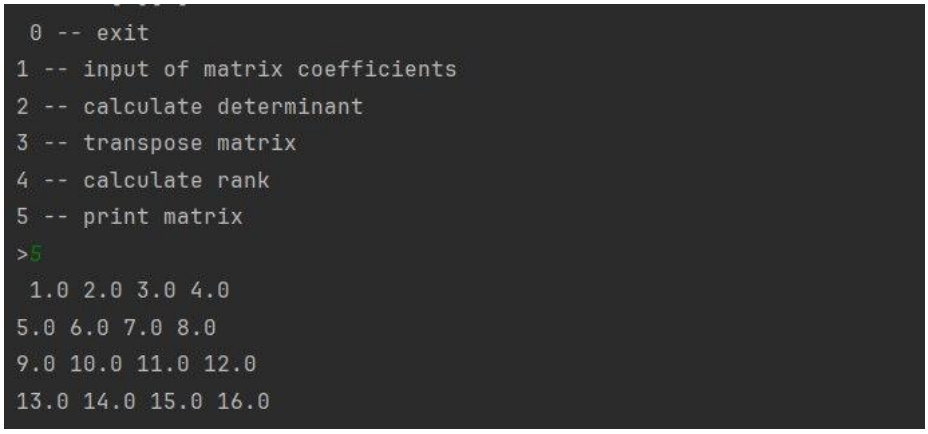


Рисунок 4 – Вывод матрицы.

После того, как у нас матрица задана, мы производим над ней вычисления. Сначала вычислим транспонированную матрицу (рисунок 4).

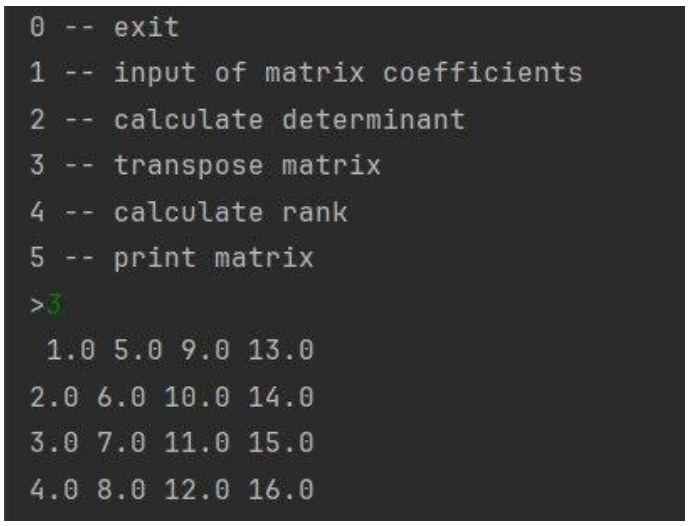


Рисунок 5 – Транспонирование матрицы.

Далее вычисляем ранг матрицы (рисунок 6).

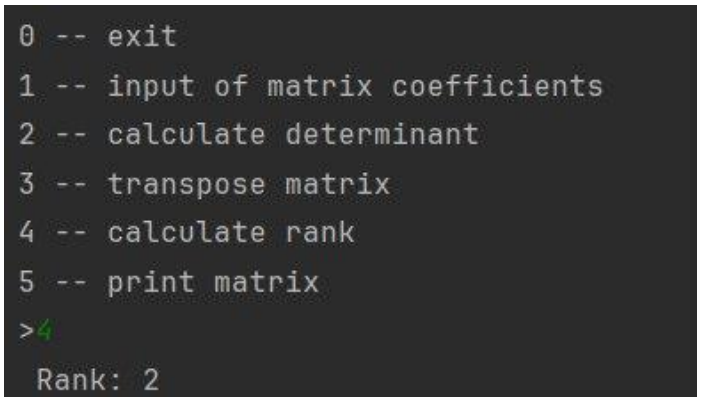


Рисунок 6 – Ранг матрицы.

Если же выбрать в меню пункт под номером 2, то программа вычислит определитель матрицы (рисунок 7).

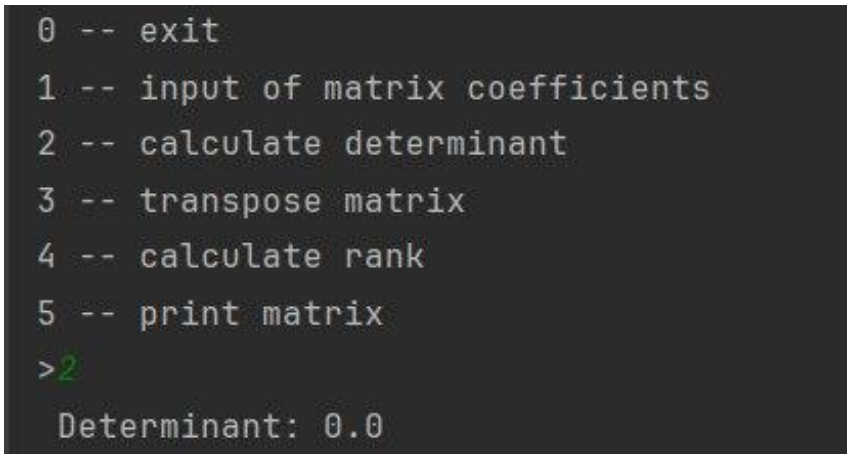


Рисунок 7 – Определитель матрицы.

После всех нужных вычислений, с помощью пункта под номером 0, мы можем завершить работу программы (рисунок 8).

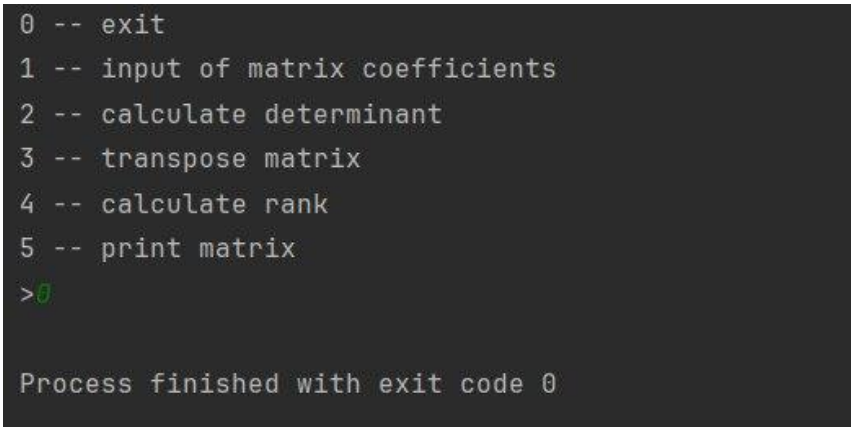


Рисунок 8 – Завершение работы программы.

# ВЫВОДЫ ПО ВЫПОЛНЕННОЙ РАБОТЕ

В ходе выполнения данной практической работы на языке C++ было разработано консольное приложение, которое выполняет вычисления с использованием квадратной матрицы, элементы которой являются вещественными числами.

В ходе выполнения работы были реализованы два класса:

- ConsoleApplication
- SquareMatrix

В результате работы приложения были получены расчеты матрицы. Эти расчеты полностью совпали с теоретическими расчетами, проведенными ранее.