**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №2**

**по дисциплине «Объектно-Ориентированное Программирование»**

|  |  |  |
| --- | --- | --- |
| Студенты группы № 2372 |  | Полуянов В.Н., Юрин А.А. |
| Преподаватель |  | Егоров С.С. |

Санкт-Петербург

2023

**Задание**

Создать консольное приложение согласно представленной на рис.1 диаграмме классов, предназначенное для заданных вычислений над квадратной матрицей, заданной на *множестве вещественных чисел*. Для этого необходимо специфицировать пользовательские классы "Консольное приложение" и "Квадратная матрица", то есть 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.

Приложение должно включать основной модуль (функция *main*), модуль «*application*» и модуль «*matrix*».

В основном модуле консольного приложения должен создаваться объект класса "Консольное приложение" и вызываться его метод, который предоставляет пользователю меню команд приложения.

Модуль **«***application***»** должен содержать спецификацию класса "Консольное приложение" и реализацию его методов. Один из его методов должен выводить в консоль меню команд приложения, включающее:

- команду, инициирующую ввод с консоли значений, задающих объект матрицы (до ввода в программе должна быть задана матрица по умолчанию);

- команду, инициирующую расчет определителя матрицы и вывод результатов расчета;

- команду, инициирующую формирования транспонированной матрицы и ее вывода в консоль;

- команду, инициирующую расчет ранга матрицы и вывод результатов расчета;

- команду, инициирующую представление в консоль текущего объекта матрицы;

- команду выхода из приложения.

Модуль **«***matrix***»** должен содержать спецификацию класса "Квадратная матрица" и реализацию его методов, необходимых для достижения цели разрабатываемого приложения. Описание класса должно использовать вместо типа double (вещественное число, заданное в условии) абстрактный тип *number*, описание которого должно задаваться в отдельном заголовочном файле *number.h* с помощью оператора *typedef double number*(для ЯП С++).

Основное требование к реализации класса **«***matrix***»** заключается в том, что она должна быть инвариантна (одна и та же для различных вариантов использования) ко множеству применимых объектов. Здесь задано множество определения элементов матрицы как вещественное, а может быть и множества целых, комплексных, рациональных и т.д. Для различных множеств при сохранении функциональности реализация класса должна быть одна и та же. Если для различных множеств менять реализацию, то теряется смысл ООП как парадигмы.

Требуется реализовать и отладить программу, удовлетворяющую сформулированным требованиям и заявленным целям. Разработать контрольные примеры и оттестировать на них программу. Оформить отчет, сделать выводы по работе.

**Спецификации классов**

**Class TApplication**

Атрибуты:

Ничего

Методы:

+TApplication() – конструктор класса

+execute() – запускает основной цикл программы, возвращает 0 в случае успешного завершения программы

-menu(char&) – выводит меню и ожидает ввода пользователя, обновляет переменную, переданную по ссылке и возвращает true если ввод успешен иначе false

**Class TMatrix**

Атрибуты:

-unsigned short size – определяет размер заданной матрицы

-std::vector<std::vector<number>>values - определяет значения матрицы

Методы:

+TMatrix() – конструктор класса без предустановленного размера

+TMatrix(unsigned short) – конструктор с предустановленным размером

+setValues(std::vector<std::vector<number>>&) – изменяет значения матрицы

+setSize(unsigned short) – изменяет размер матрицы (расширяет или сжимает её с сохранением значений, новые - нули)

+getSize() – возвращает unsigned short текущий размер матрицы

+operator<<(std::ostream&, TMatrix&) – выводит матрицу в поток вывода

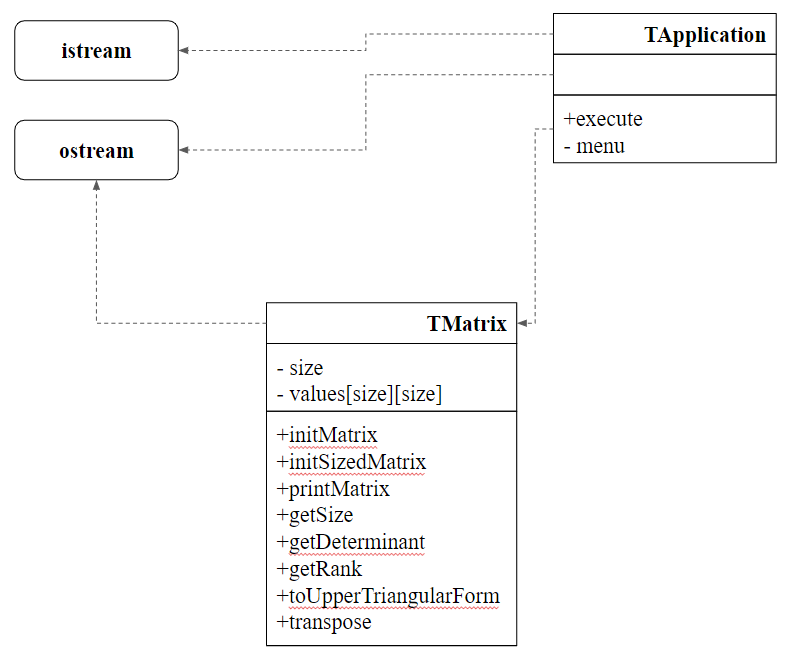
+getDeterminant() – возвращает number определитель матрицы

+getRank() – возвращает unsigned int ранг матрицы

+transpose() – транспонирует матрицу

#toUpperTriangularForm() – приводит матрицу к верхнему ступенчатому виду с помощью метода Гаусса. Используется для вычисления определителя и ранга матрицы.

**Диаграмма классов, дополненная атрибутами и методами**

****

**Описание контрольного примера с исходными и ожидаемыми (расчетными) данными**

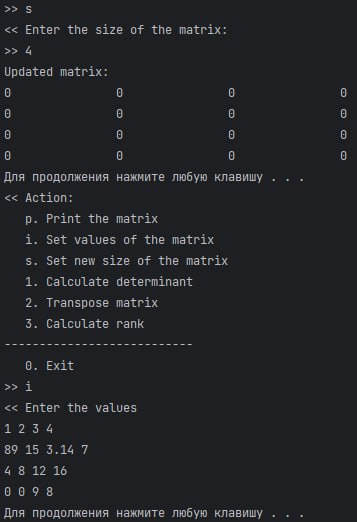
|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 89 | 15 | 3.14 | 7 |
| 4 | 8 | 12 | 16 |
| 0 | 0 | 9 | 8 |

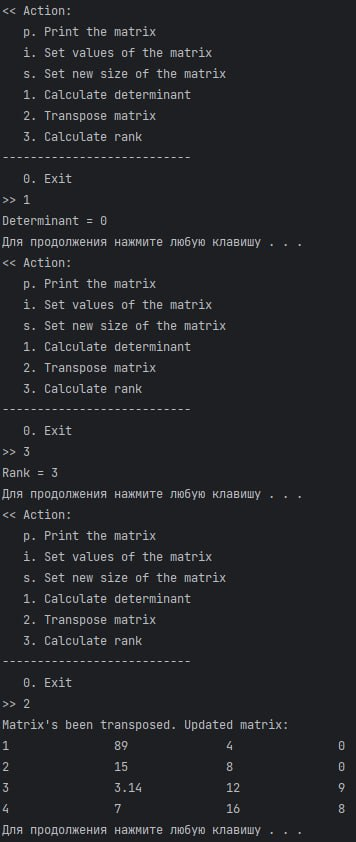
Определитель такой матрицы будет равняться 0, а ранг равняться 3. При транспонировании мы получим матрицу:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 89 | 4 | 0 |
| 2 | 15 | 8 | 0 |
| 3 | 3.14 | 12 | 9 |
| 4 | 7 | 16 | 8 |

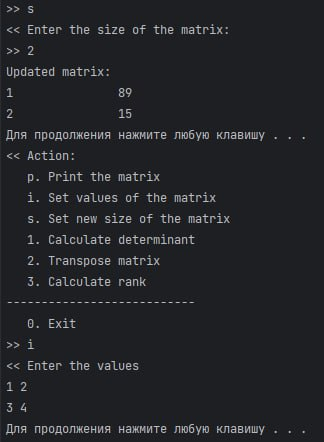
**Скриншоты работы программы на контрольных примерах. Два примера**

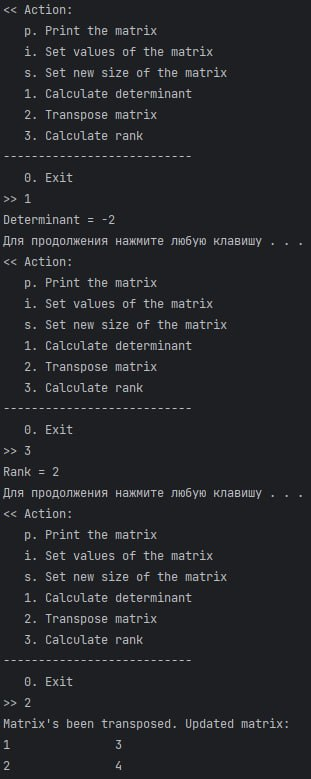
1. *Первый пример*





1. *Второй пример*





**Вывод**

В данной практической работе было успешно разработано консольное приложение для выполнения вычислений над матрицей вещественных чисел. Это приложение может быть дополнено и расширено для работы с различными типами данных, включая числа из других множеств.