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

Кафедра информационных систем

ОТЧЕТ

по практической работе №3

по дисциплине «Объектно-ориентированное программирование»

Тема: GUI приложение для вычислений над квадратной матрицей на множестве рациональных чисел

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

Преподаватель _____ Егоров С.С.

Санкт-Петербург
2024

ЗАДАНИЕ НА ПРАКТИЧЕСКУЮ РАБОТУ

Тема работы: GUI приложение, реализующее функции на множестве рациональных чисел.

Исходные данные:

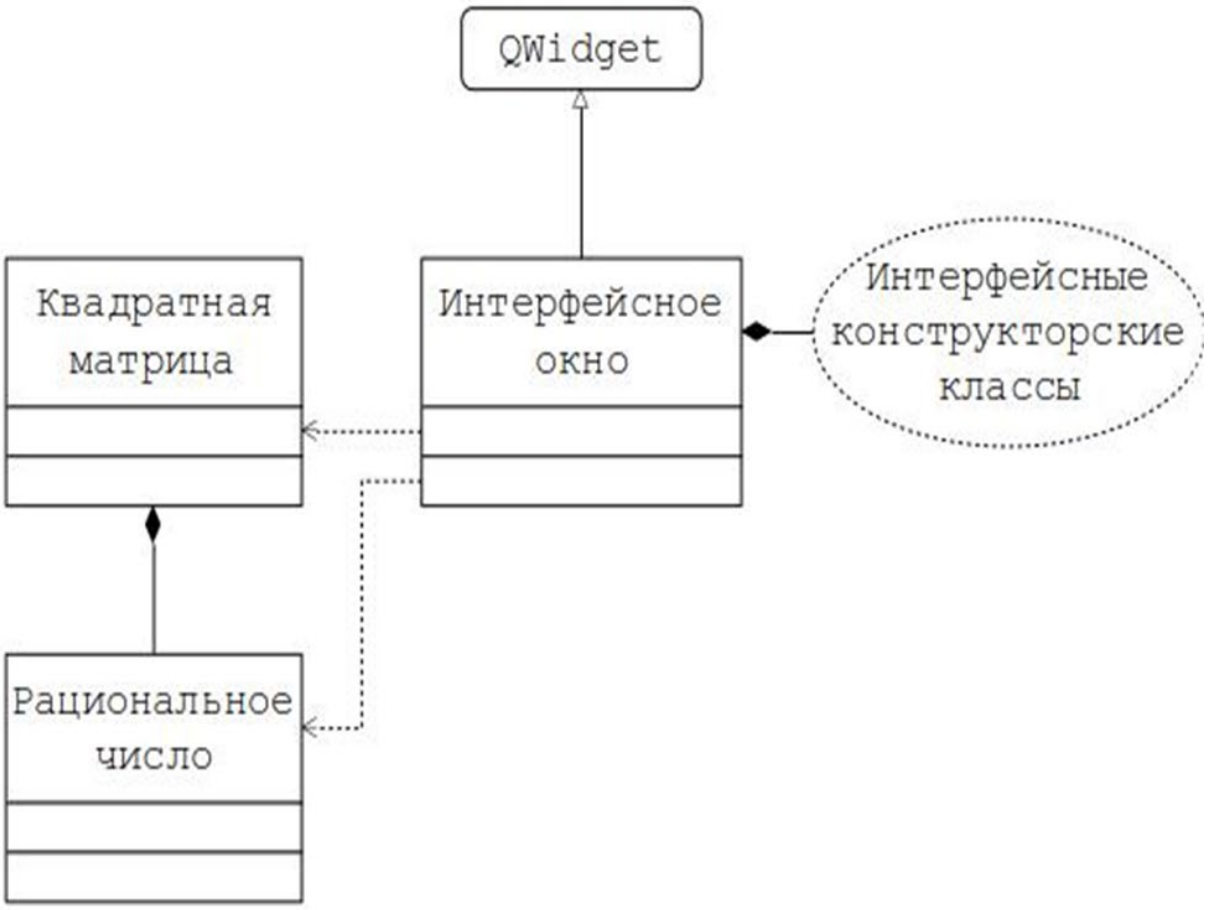

Рис.1 – Диаграмма классов

Описание работы:

Создать GUI приложение, реализующее функции перечисленные в описании работы №1, но на множестве рациональных чисел. Для этого требуется разработать и реализовать класс рациональных чисел.

Рациональное число — это несократимая дробь a/b, где a и b целые, причем b>0.

Приложение должно включать основной модуль, модуль «interface», модуль «matrix», модуль «rational» и файл number.h:

При необходимости расширения функциональности класса «Квадратная матрица» следует только дополнить его протокол без каких-либо изменений уже существовавшей реализации.

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

СОДЕРЖАНИЕ

1. СПЕЦИФИКАЦИЯ РАЗРАБОТАННЫХ КЛАССОВ

1.1. Класс `TRational` («Рациональные числа»):
Атрибуты класса:

a) `int m_numerator, m_denominator`

Область видимости: private.

Описание: хранение числителя и знаменателя дроби соответственно.

Стандартное значение: 0.

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

Методы класса:

a) `TRational()`

Область видимости: private.

Назначение: Конструктор, инициализирует объект класса TRational с заданными числителем и знаменателем, используя значения по умолчанию 0 и 1 соответственно.

b) `int numerator()`
`const`

Область видимости: private.

Назначение: возвращает числитель дроби.

c) `int denominator() const`

Область видимости: private.

Назначение: возвращает знаменатель дроби.

d) `Void normalize()`

Область видимости: private.

Назначение: нормализует дробь, упрощая её до наименьших возможных

значений числителя и знаменателя..

e) `operator+, operator- operator*, operator/, operator ==, operator!=, operator>`

Область видимости: private.

Назначение: операторы для выполнения арифметических операций и сравнения дробей.

f) `abs()`

Область видимости: private.

Назначение: статический метод, возвращающий абсолютное значение дроби.

g) `int gcd (int a, int b)`

Область видимости: private.

Назначение: статический метод, вычисляющий наибольший общий делитель двух чисел.

h) `toString()`

Область видимости: private.

Назначение: метод, возвращающий строковое представление дроби.

1.2. Класс `Widget` («Родительский класс интерфейса»):

Атрибуты класса:

Не имеется.

Методы класса:

a) `Widget(QWidget *parent = nullptr)`

Область видимости: public.

Назначение: конструктор, который инициализирует объект класса Widget, принимая указатель на родительский виджет в качестве параметра с

значением по умолчанию nullptr.

b) `~Widget()`

Область видимости: public.

Назначение: деструктор, который уничтожает объект класса Widget.

Методы имеют открытый тип доступа дабы могли использоваться вне класса.

1.3. Класс `MatrixLayoutWidget`

Атрибуты класса:

a) `Int row, int column`
Область видимости: private
Назначение: хранят в себе строки и столбцы матрицы.

b) `Matrix`
Область видимости: private
Назначение: указатель на объект класса SquareMatrix.

Методы класса:

a) `onOutputMatrixTransp()`
Область видимости: private
Назначение: Метод для вывода транспонированной матрицы..

b) `onTransposeMatrix()`
Область видимости: private
Назначение: Метод для транспонирования матрицы.

c) `onDeterminantMatrix()`
Область видимости: private
Назначение: Метод для вычисления определителя матрицы.

d) `onRangMatrix()`
Область видимости: private
Назначение: Метод для вычисления ранга матрицы.

e) `removeNumeratorDenominatorFields()`
Область видимости: private
Назначение: Метод для удаления полей числителя и знаменателя.

1.4. Класс `MatrixDisplayWindow` (дочерний класс от Widget).

Атрибуты класса:

a) `QTextEdit *matrixTextEdit`
Область видимости: private
Назначение: указатель на объект.

Методы:

a) `explicit MatrixDisplayWindow(QWidget *parent = nullptr)`
Область видимости: public
Назначение: конструктор.

b) `void displayMatrix(const std::stringstream &matrixText)`
Область видимости: public
Назначение: отображение матрицы в текстовом поле.

2. ДИАГРАММА КЛАССОВ

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

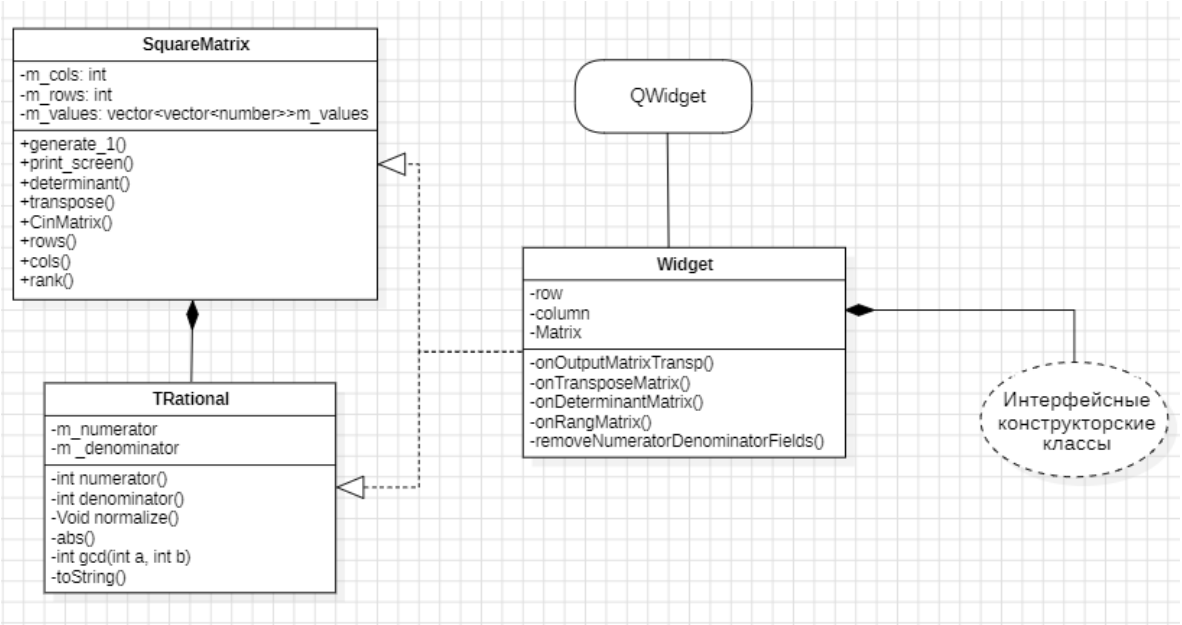

Рисунок 2 – Диаграмма классов

3. ОПИСАНИЕ КОНТРОЛЬНОГО ПРИМЕРА

Исходные данные для контрольного примера:

$$A = \begin{pmatrix} \frac{1}{2} & \frac{3}{4} \\ \frac{7}{5} & \frac{8}{3} \end{pmatrix}$$

Чтобы найти транспонированную матрицу поменяем строки столбцы матрицы местами:

$$\mathbf{A^T} = \begin{pmatrix} \frac{1}{2} & \frac{7}{5} \\ \frac{3}{4} & \frac{8}{3} \end{pmatrix}$$

Далее ищем определитель матрицы:

$$\det A = \begin{vmatrix} \frac{1}{2} & \frac{3}{4} \\ \frac{7}{5} & \frac{8}{3} \end{vmatrix} = \frac{1}{2} * \frac{8}{3} - \frac{7}{5} * \frac{3}{4} = \frac{4}{3} - \frac{21}{20} = \frac{17}{60}$$

Т.к. det $A \neq 0$, то линейно зависимых строк или столбцов нет $\Rightarrow$ rank A = 2.

4. РАБОТА ПРОГРАММЫ НА КОНТРОЛЬНЫХ ПРИМЕРАХ

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

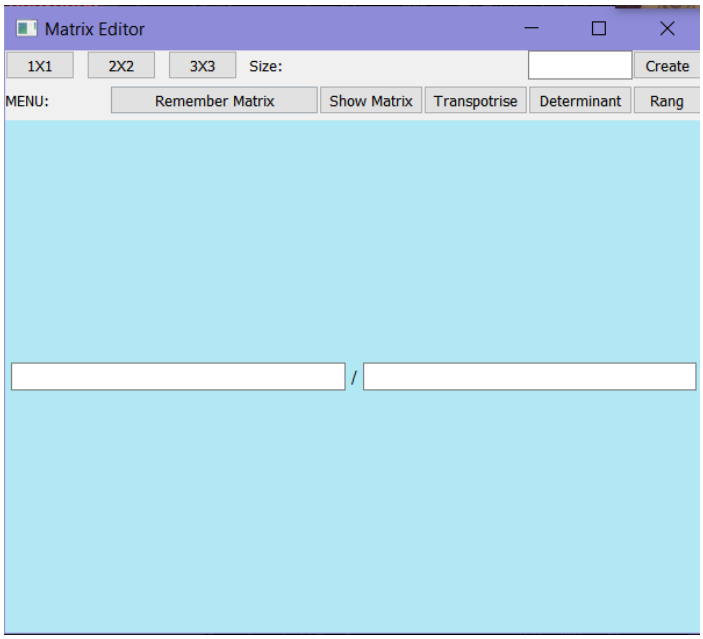

Рисунок 3 – Меню

Далее производим ввод пользовательской матрицы в приложении.

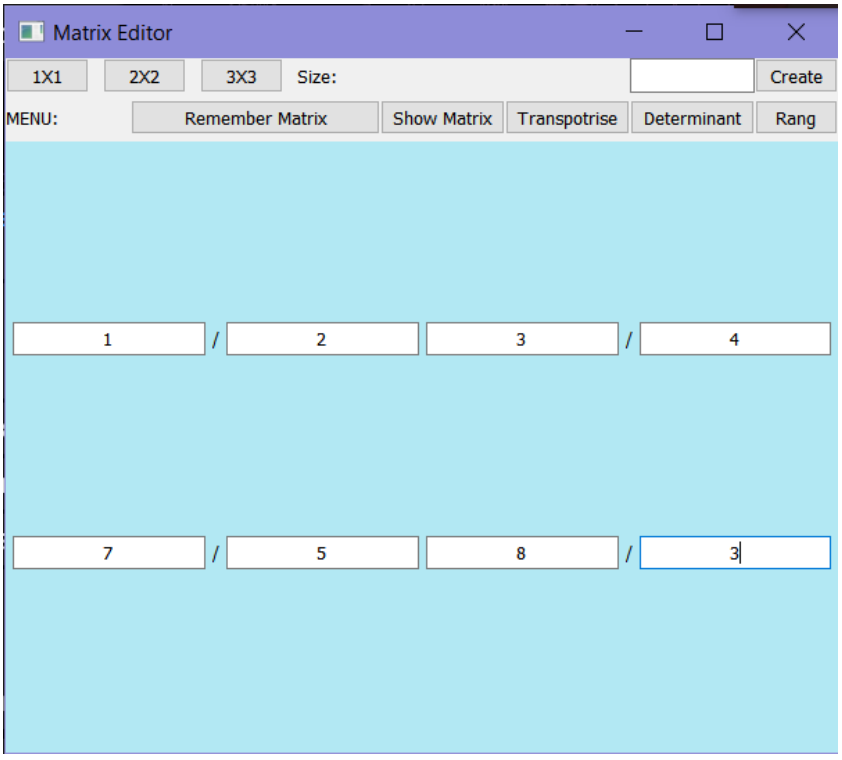

Рисунок 4 – Ввод пользовательской матрицы

После того как мы ввели коэффициенты, можем, нажав кнопку “Remember

Matrix”, записать данную матрицу в память. Приложение выведет следующее сообщение (рисунок 5).

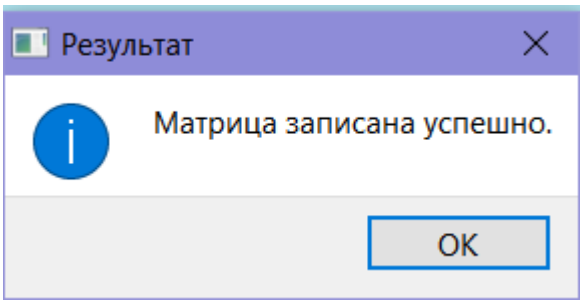

Рисунок 5 – Результат кнопки «Remember Matrix»

После того мы определили матрицу, ее можно вывести, нажав кнопку «Show Matrix» (рисунок 6).

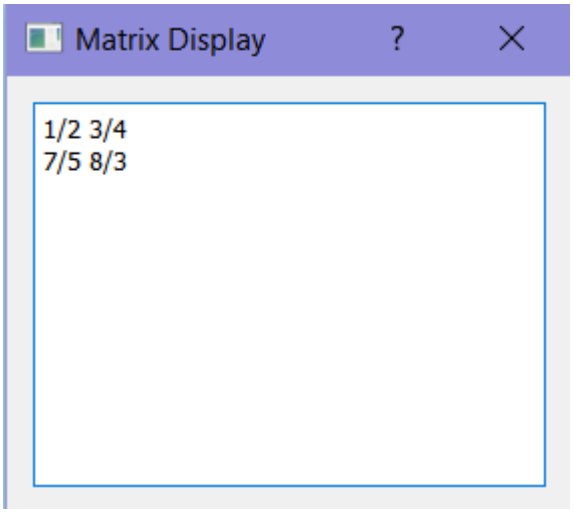

Рисунок 6 – Вывод матрицы.

После того, как у нас матрица задана, мы производим над ней вычисления. Сначала вычислим транспонированную матрицу, нажав кнопку“Transpotrise” (рисунок 7).

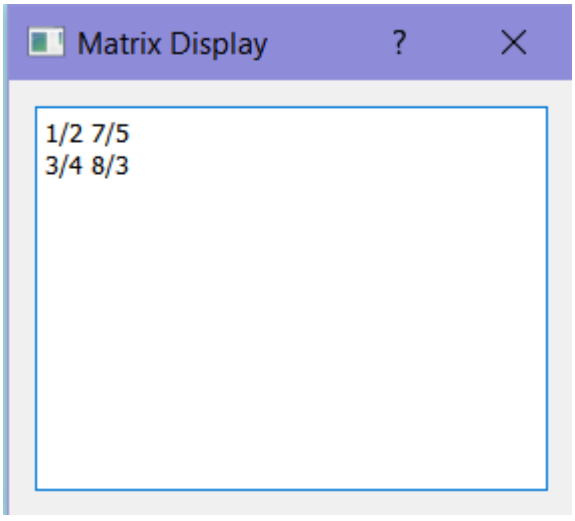

Рисунок 7 – Транспонирование матрицы.

Далее вычисляем ранг матрицы, нажав кнопку “Rang” (рисунок 8).

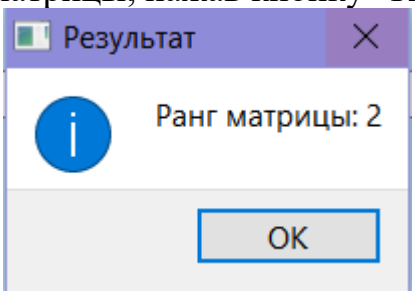

Рисунок 8 – Ранг матрицы.

Если же выбрать кнопку “Determinant, то программа вычислит определитель матрицы (рисунок 9).

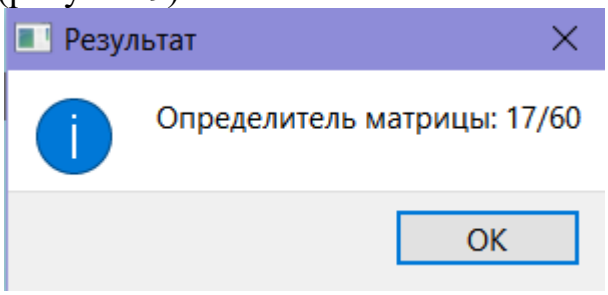

Рисунок 9 – Определитель матрицы.

Для завершения программы достаточно нажать на крест в верхнем правом углу.

ВЫВОДЫ ПО ВЫПОЛНЕННОЙ РАБОТЕ

Разработано GUI приложение для работы с рациональными числами, включающее модули для основных операций и графического интерфейса. Приложение было структурировано с использованием модульной архитектуры, что обеспечило гибкость и расширяемость. Программа была протестирована на контрольных примерах, что подтвердило ее надежность и эффективность.

Выводы:

1. Разработка GUI приложения для работы с рациональными числами расширяет возможности программирования и анализа данных.
2. Использование модульной архитектуры обеспечивает гибкость и расширяемость приложения.
3. Необходимость тестирования подчеркивает важность качественного программного обеспечения.
4. В дальнейшем, расширение функциональности класса "Квадратная матрица" может быть выполнено без изменения существующей реализации.

В целом, работа над проектом позволила углубить понимание принципов разработки GUI приложений и работы с рациональными числами, а также практически применить полученные знания в реальном проекте.