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»
Тема: Консольное приложение для вычислений над квадратной матрицей, заданной комплексными 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, реализующее функции перечисленные в описании работы №1, но на множестве комплексных чисел.

Исходные данные:

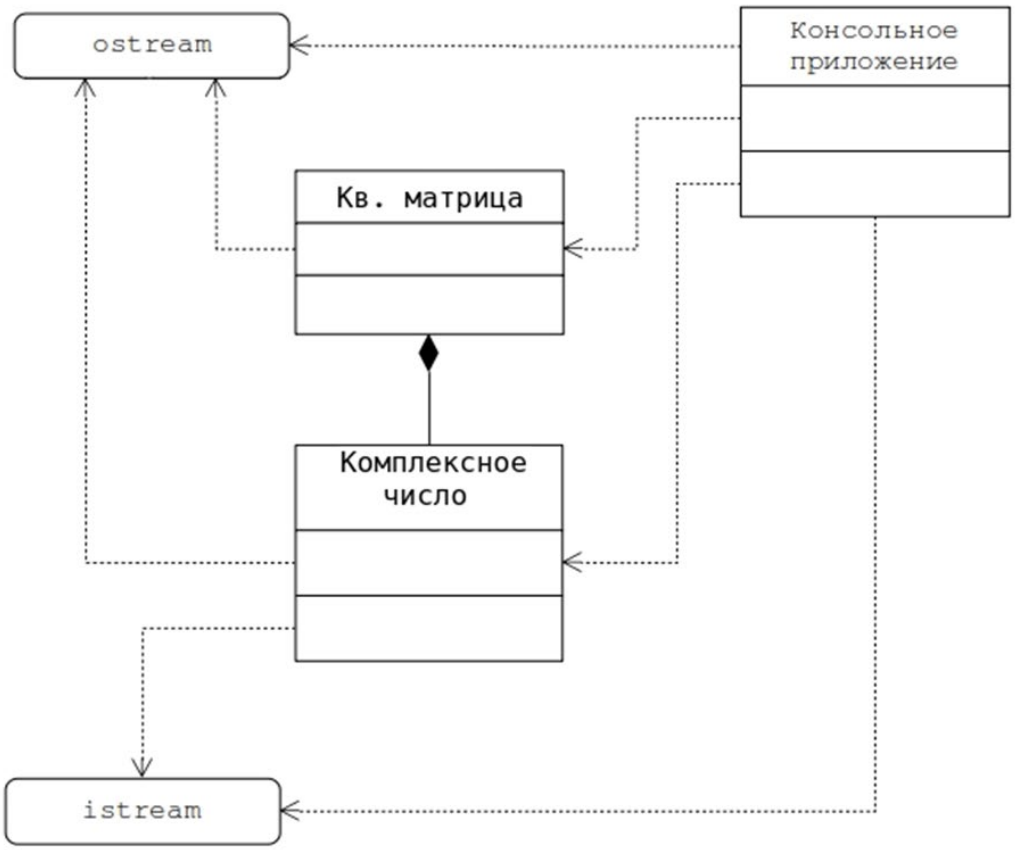

Рис.1 – Диаграмма классов

Описание работы:

Создать консольное приложение, реализующее функции перечисленные в описании работы №1, но на множестве комплексных чисел.

Приложение должно включать основной модуль, модуль «application», модуль «matrix» и модуль «complex. 3

Для этого в проект лабораторной работы №1 следует добавить модуль с собственным (не брать из ООБ!) описанием и реализацией класса комплексных

чисел TComplex. Класс TComplex должен быть встроен в проект согласно диаграмме классов на рис.1. При этом основной модуль, модуль «application» и модуль «matrix» не должны изменяться.

В классе TComplex следует определить только те члены класса и спецификации, которые необходимы для совместимости модулей проекта и реализации отношений, приведенных в ДК объектной модели.

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

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) `double 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) `double determinantHelper (const std:: vector<std:: vector<double>>& 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`.

1.3. Класс `TComplex` («Комплексные числа»):

Атрибуты класса:

a) `double real;`

Область видимости: private.

Описание: действительная часть комплексного числа.

Стандартное значение: real = 0.

b) `double imaginary;`

Область видимости: private.

Описание: мнимая часть комплексного числа.

Стандартное значение: imaginary = 0.

Для защиты от неконтролируемого доступа, атрибуты класса имеют закрытый тип доступа, они используются только внутри модуля, где описан класс комплексного числа.

Методы класса:

a) `TComplex ();`

Область видимости: public.

Назначение: конструктор комплексного числа по умолчанию.

b) `TComplex(const int&);`

Область видимости: public.

Назначение: конструктор комплексного числа по заданному числу. Нужен для создания пользовательской матрицы квадратного типа.

c) `~TComplex ();`

Область видимости: public.

Назначение: Деструктор комплексного числа.

d) `ostream& operator<< (ostream& os, TComplex c);`

Область видимости: public.

Назначение: вывод на экран комплексного числа. Возвращает указатель на поток вывода.

e) `istream& operator>> (istream& is, TComplex& c);`

Область видимости: public.

Назначение: ввод на экран комплексного числа. Возвращает указатель на поток ввода.

f) `TComplex abs (const TComplex& t);`

Область видимости: public.

Назначение: вычисление модуля комплексного числа. Возвращает вещественное значения модуля комплексного числа.

g) `TComplex operator*(int n, const TComplex& c);`

Область видимости: public.

Назначение: умножение целого числа на матрицу комплексных чисел. Возвращает матрицу комплексных чисел

h) `double abs () const;`

Область видимости: public.

Назначение: нужен для вычисления модуля сиюминутного комплексного числа. Вспомогательная функция. Возвращает вещественное значения модуля комплексного числа.

i) `TComplex operator*(TComplex) const;`

Область видимости: public.

Назначение: перемножение матриц над комплексными числами. Возвращает матрицу результата.

j) `TComplex operator*(int n) const;`

Область видимости: public.

Назначение: умножение целого числа на комплексное число. Возвращает преобразованное комплексное число.

k) `TComplex operator- (const TComplex& other) const;`

Область видимости: public.

Назначение: возвращает разность двух комплексных чисел.

l) `TComplex operator+ (const TComplex& other) const;`

Область видимости: public.

Назначение: возвращает сумму двух комплексных чисел.

m) `TComplex operator/ (TComplex& other);`

Область видимости: public.

Назначение: возвращает частное двух комплексных чисел.

n) `bool operator> (const TComplex& other);`

Область видимости: public.

Назначение: возвращает true если комплексное число больше, иначе false.

o) `bool operator== (TComplex);`

Область видимости: public.

Назначение: возвращает true если комплексные числа равны, иначе false.

p) `TComplex operator/= ( TComplex& other);`

Область видимости: public.

Назначение: возвращает матрицу равную частному этой же матрицы и числа.

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.

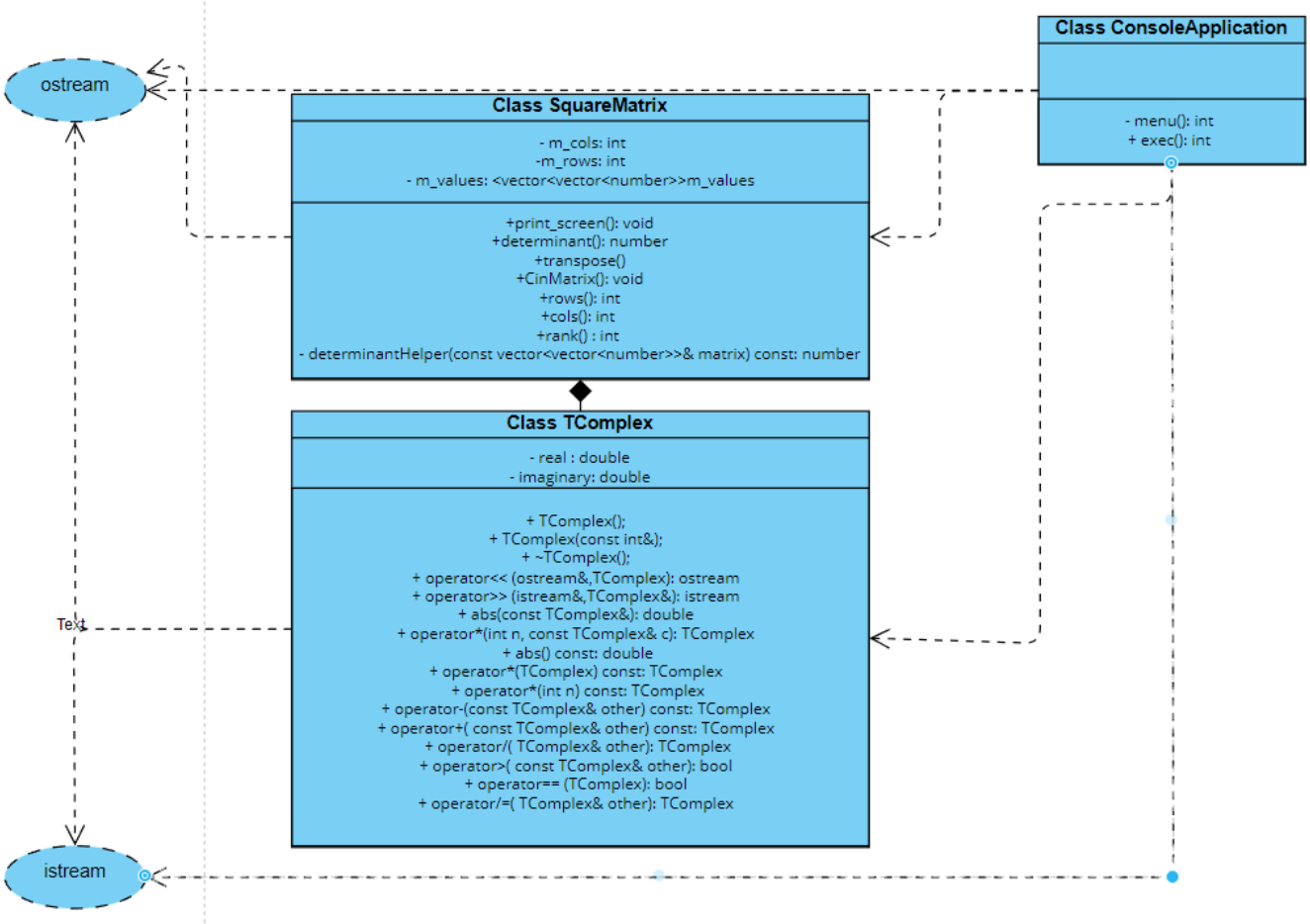

Рисунок 2 – Диаграмма классов

3. ОПИСАНИЕ КОНТРОЛЬНОГО ПРИМЕРА

Посчитаем ранг для данной матрицы.

$$A = \begin{bmatrix} 3+8i & 4+6i \\ 2+8i & 3-9i \end{bmatrix}$$

Приведем матрицу к верхнетреугольной матрицей:

Разделим первую строку на (3+8i).

$$A = \begin{bmatrix} 1 & (4+6i)/(3+8i) \\ 2+8i & (3-9i) \end{bmatrix}$$

Умножаем первую строку на (2+8i).

$$A = \begin{bmatrix} 2+8i & (-40+44i)/(3+8i) \\ 2+8i & (3-9i) \end{bmatrix}$$

Вычитаем первую строку из второй и записываем результат во вторую строку.

$$A = \begin{bmatrix} 1 & (4+6i)/(3+8i) \\ 0 & (121-47i)/(3+8i) \end{bmatrix}$$

Восстановим первую строку матрицы до изначального состояния.

$$A = \begin{bmatrix} 3+8i & 4+6i \\ 0 & (121-47i)/(3+8i) \end{bmatrix}$$

Таким образом количество линейно независимых строк =2. Следовательно ранг матрицы равен 2.

Посчитаем определитель для данной матрицы.

$$A = \begin{bmatrix} 3+8i & 4+6i \\ 2+8i & 3-9i \end{bmatrix}$$

Воспользуемся преобразованием которые проведены выше:

$$A = \begin{bmatrix} 3+8i & 4+6i \\ 0 & (121-47i)/(3+8i) \end{bmatrix}$$

Определитель считается по следующей формуле:

$$\Delta = (3+8i) * (121-47i) \div (3+8i) = (121-47i)$$

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.

```
0 -- exit
1 -- input of matrix coefficients
2 -- calculate determinant
3 -- transpose matrix
4 -- calculate rank
5 -- print matrix
>1
 enter matrix dimensions
n =2
 enter coefficient matrices

matrix [0][0] =3 8

matrix [0][1] =4 6

matrix [1][0] =2 8

matrix [1][1] =3 -9
```

Рисунок 4 – ввод коэффициентов матрицы

После того мы определили матрицу, ее можно вывести (рисунок 5).

```
 0 -- exit
1 -- input of matrix coefficients
2 -- calculate determinant
3 -- transpose matrix
4 -- calculate rank
5 -- print matrix
>5
 (3.0+8.0i) (4.0+6.0i)
(2.0+8.0i) (3.0+-9.0i)
```

Рисунок 5 – Вывод матрицы.

После того, как у нас матрица задана, мы производим над ней вычисления. Сначала вычислим транспонированную матрицу (рисунок 6).

```
0 -- exit
1 -- input of matrix coefficients
2 -- calculate determinant
3 -- transpose matrix
4 -- calculate rank
5 -- print matrix
>3
 (3.0+8.0i) (2.0+8.0i)
(4.0+6.0i) (3.0+-9.0i)
```

Рисунок 6 – Транспонирование матрицы.

Далее вычисляем ранг матрицы (рисунок 7).

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

Рисунок 7 – Ранг матрицы.

Если же выбрать в меню пункт под номером 2, то программа вычислит определитель матрицы (рисунок 8).

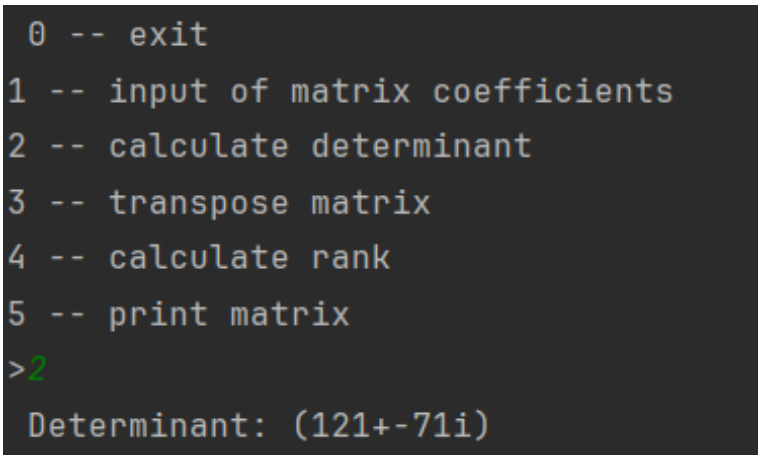

Рисунок 8 – Определитель матрицы.

После всех нужных вычислений, с помощью пункта под номером 0, мы можем завершить работу программы (рисунок 9).

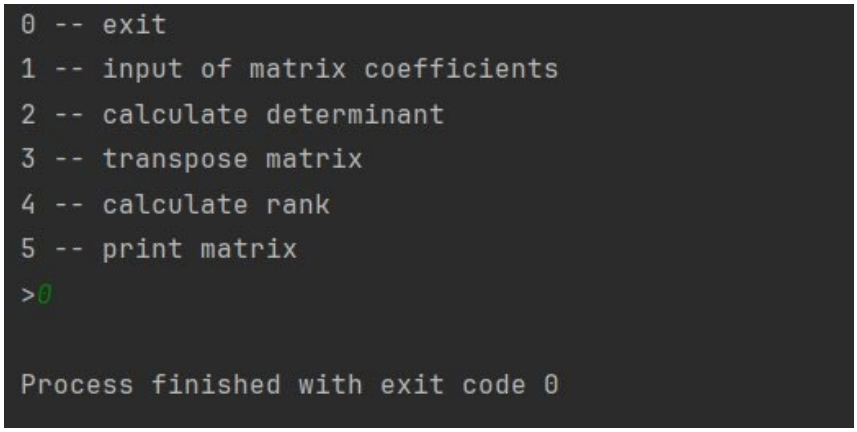

Рисунок 9 – Завершение работы программы.

ВЫВОДЫ ПО ВЫПОЛНЕННОЙ РАБОТЕ

В ходе выполнения лабораторной работы было создано консольное приложение, реализующее функции для матриц на множестве комплексных чисел. Приложение было построено на основе модульной структуры, включающей основной модуль, модуль «application», модуль «matrix» и дополнительный модуль «complex», содержащий класс TComplex.

Класс TComplex был разработан с учетом требований к совместимости с остальными модулями проекта и реализации отношений, приведенных в диаграмме классов. В классе TComplex были определены только необходимые члены класса и спецификации, что позволило обеспечить гибкость и модульность приложения.

В процессе разработки и отладки программы были учтены все требования и цели, заявленные в описании работы. Были разработаны контрольные примеры, на которых была протестирована программа, что позволило убедиться в ее корректной работе.

В ходе работы были выявлены и устранены некоторые технические проблемы, связанные с обработкой ввода данных и выводом результатов. Также были исправлены некоторые ошибки, допущенные в предыдущей работе (в частности, две размерности одной матрицы). Это позволило улучшить качество программы и сделать ее более надежной и удобной для пользователя.

В результате работы была получена работающая программа, способная выполнять заданные функции на множестве комплексных чисел. Программа была тщательно протестирована, что подтвердило ее корректность и эффективность.

В дальнейшем планируется расширить функциональность программы, добавив новые функции и улучшив интерфейс пользователя. Также предполагается исследовать возможности оптимизации производительности программы и улучшения ее масштабируемости.