Федеральное государственное автономное образовательное учреждение

высшего образования «Санкт-Петербургский государственный

электротехнический университет «ЛЭТИ»

им. В.И. Ульянова (Ленина)»

**Кафедра информационных систем**

**ОТЧЕТ**

**по практической работе №2**

**по дисциплине «Объектно-ориентированное программирование»**

Студент гр. №3376 Михайлов Н.  
Студентка гр. №3376 Дегтярева М.  
  
Преподаватель Егоров С.С.

Санкт-Петербург  
2024

Задание на практическую работу:   
Создать консольное приложение предназначенное для работы с массивом вещественных и комплексных чисел.  
Приложение должно включать основной модуль, модуль «application», модуль «array» и модуль «complex».   
Написать собственную реализацию класса комплексных чисел.  
Приложение должно работать с разработанным классом “Массив” (Array) способным: создавать 0 массив, создавать массив с N элементами и заполнять их заданными значениями, корректно уничтожать объекты класса, изменять размерность массива, вводить элементы массива из istream, выводить элементы в ostream, вычислять среднее отклонение и СКО впо формулам, сортировать элементы массива по возрастанию и убыванию.  
Так же разработать контрольные примеры и оттестировать на них программу. Оформить отчет, сделать выводы по работе.

Таблица 1. Первичный протокол класса APPLICATION

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибуты (старые) | | | | |
| идентификатор | тип | | Область видимости | семантическое описание |
| mainMenuItems | std::vector<std::string> | | private | Пункты основного меню |
| sortMenuItems | std::vector<std::string> | | private | Пункты для меню сортировки |
| arrow | std::string | | private | Стрелка |
| arrowPoss | unsigned | | private | Для отслеживания позиции стрелки |
| Методы (старые) | | | | |
| идентификатор | | тип | Область видимости | семантическое описание |
| Application(); | | - | public | Конструктор |
| ~Application() | | - | public | деструктор |
| MainMenu(); | | Void | public | Вывод основного меню |
| SortMenu(); | | Void | public | Вывод меню сортировки |
| GetArrowPoss() | | unsigned | public | Получение позиции стрелки |
| SetArrowPoss(unsigned t\_arrowPoss) | | void | public | Установка позиции стрелки |

Таблица 2. Первичный протокол класса ARRAY

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Атрибуты (старые) | | | | | | |
| идентификатор | тип | | область видимости | | семантическое описание | |
| array | std::vector<number> | | private | | Задающийся массив | |
| Методы (старые) | | | | | | |
| идентификатор | | тип | | область видимости | | семантическое описание |
| Array() | | - | | public | | Конструктор |
| ~ Array() | | - | | public | | Деструктор |
| Create(); | | bool | | public | | Создание пустого массива |
| Sort(bool ascending = true); | | bool | | public | | Сортировка массива |
| SetValue(unsigned t\_index, number value); | | bool | | public | | Изменение значений элементов массива |
| ChangeSize(unsigned t\_size); | | bool | | public | | Изменение размера массива |
| Print(); | | bool | | public | | Вывод массива |
| Mean(); | | number | | public | | Расчет среднего арифметического значения |
| StandardDeviation(); | | number | | public | | Расчет СКО |
| GetArray() | | std::vector<number> | | public | | Получение массива |

Таблица 2. Первичный протокол класса COMPLEX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибуты | | | | |
| идентификатор | тип | область видимости | | семантическое описание |
| real | double | private | | Действительная часть комплексного числе |
| Imaginary | double | private | | Мнимая часть комплексного числа |
| Методы | | | | |
| идентификатор | тип | | область видимости | семантическое описание |
| Complex() | Complex | | public | Конструктор по умолчанию, инициализирует real и imaginary значением 0. |
| Complex(double real, double imaginary) | Complex | | public | Конструктор, инициализирующий real и imaginary заданными значениями |
| getReal() const | double | | public | Возвращает действительную часть комплексного числа. |
| getImaginary() const | double | | public | Возвращает мнимую часть комплексного числа. |
| setRealPart(double real) | void | | public | Устанавливает значение действительной части. |
| setImaginaryPart(double imaginary) | void | | public | Устанавливает значение мнимой части. |
| Magnitude() const | double | | public | Вычисляет и возвращает модуль комплексного числа. |
| operator+=(const Complex& other) | Complex& | | public | Перегруженный оператор += для сложения с другим комплексным числом. |
| operator-=(const Complex& other) | Complex& | | public | Перегруженный оператор -= для вычитания другого комплексного числа. |
| operator\*=(const Complex& other) | Complex& | | public | Перегруженный оператор \*= для умножения на другое комплексное число. |
| operator/=(const Complex& other) | Complex& | | public | Перегруженный оператор /= для деления на другое комплексное число. |
| operator+(const Complex& other) const | Complex | | public | Перегруженный оператор + для сложения с другим комплексным числом и возвращения результата. |
| operator-(const Complex& other) const | Complex | | public | |  | | --- | | Перегруженный оператор - для вычитания другого комплексного числа и возвращения результата. |  |  | | --- | |  | |
| operator\*(const Complex& other) const | Complex | | public | |  | | --- | | Перегруженный оператор \* для умножения на другое комплексное число и возвращения результата. |  |  | | --- | |  | |
| operator/(const Complex& other) const | Complex | | public | Перегруженный оператор / для деления на другое комплексное число и возвращения результата. |
| operator/(double value) const | Complex | | public | Перегруженный оператор / для деления на число(double) и возвращения результата. |
| operator==(const Complex& other) const | bool | | public | Перегруженный оператор == для проверки равенства двух комплексных чисел. |
| operator!=(const Complex& other) const | bool | | public | Перегруженный оператор != для проверки неравенства двух комплексных чисел. |
| operator>(const Complex& lhs, const Complex& rhs); | bool | | public | Перегруженный оператор > для сравнения двух комплексных чисел. |
| operator<(const Complex& lhs, const Complex& rhs); | bool | | public | Перегруженный оператор < для сравнения двух комплексных чисел. |
| operator<<(std::ostream& outputStream, const Complex& number) | std::ostream& | | public | Перегруженный оператор << для вывода комплексного числа в поток. |
| operator>>(std::istream& inputStream, const Complex& number) | std::istream& | | public | Перегруженный оператор >> для ввода комплексного числа из потока. |
| GetComplexString() const | std::string | | public | Возвращает строковое представление комплексного числа. |

Диаграмма классов программы:

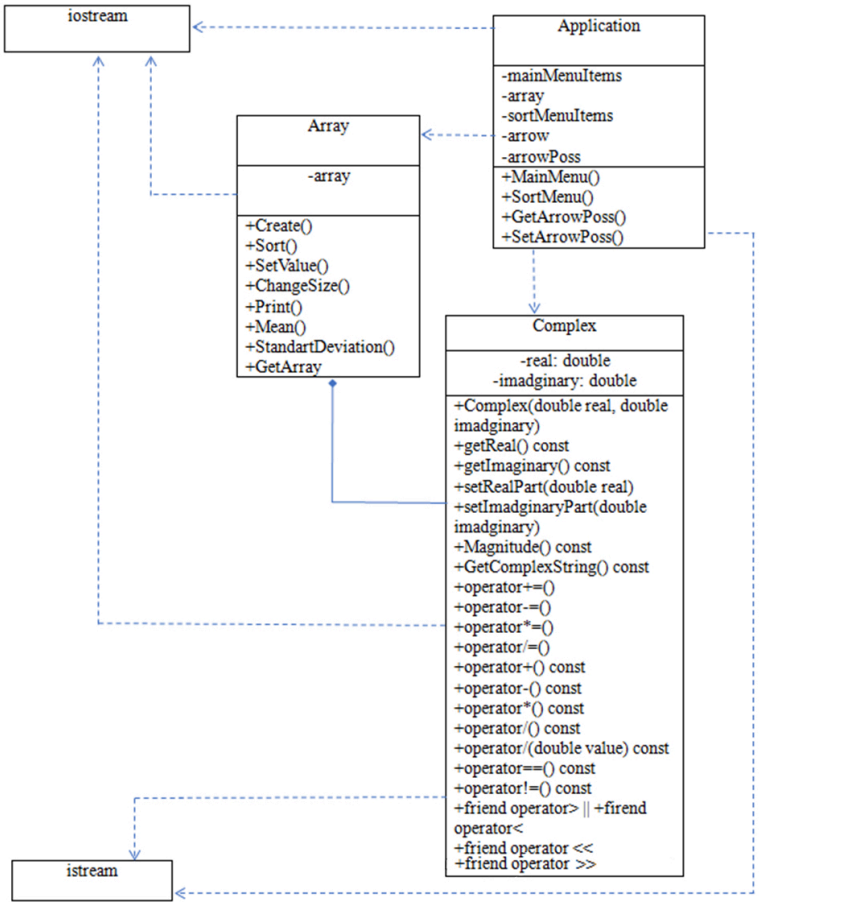


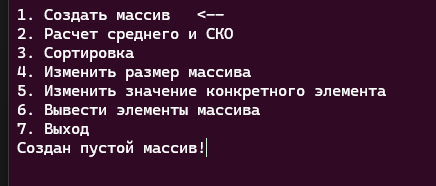
Рис.1. Диаграмма классов работы №2

Тесты работы программы:   
Контрольный пример.   
«10+2i, -3-2.3i, 1.5-13i, 5+6i, -10+14i»

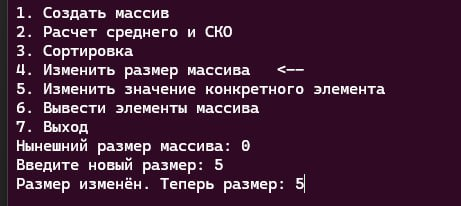
Проверим правильность работы программы на контрольном примере  
Расчетные результаты сортировки по возрастанию:  
«-3-2.3i, 5+6, 10+2i, 1.5-13i, -10+14i»  
по убыванию:  
«-10+14i, 1.5-13i, 10+2i, 5+6i, -3-2.3i»  
Расчетные результаты среднего значения и СКО:

|  |  |
| --- | --- |
|  | 0.7+1.34i |
| СКО | 11.2704+0i |

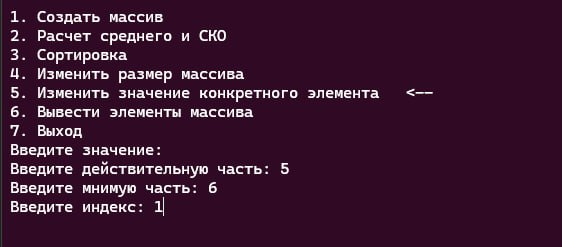
1. Создадим пустой массив



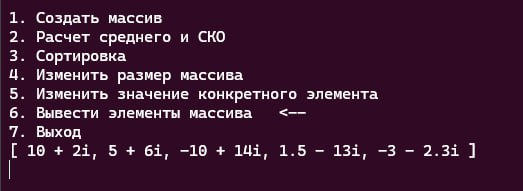
1. Изменим размер массива на нужный



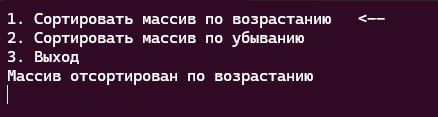
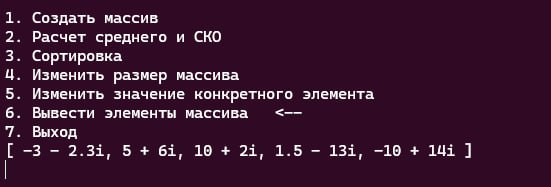
1. Изменим числа в массиве на числа из контрольного примера



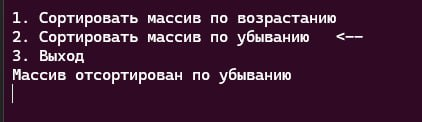
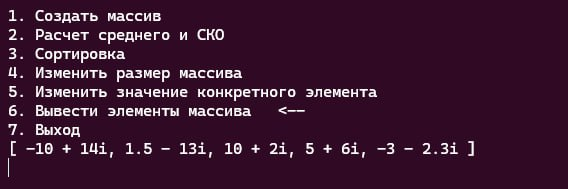
1. Проверим массив



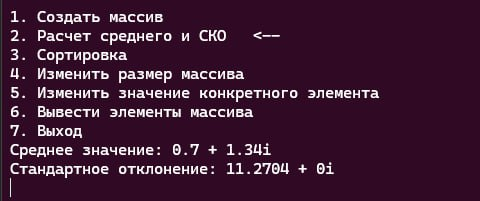
1. Отсортируем массив по возрастанию

1. Отсортируем массив по убыванию

1. Рассчитаем среднее значение и СКО по формуле, проверим совпадает ли с ожидаемым результатом.



Расчетные результаты совпадают с полученными. Программа работает корректно.

**Выводы:**   
В ходе работы было создано консольное приложение для работы с массивом комплексных чисел, способное: создавать массив с N элементами включая 0 и заполнять его заданными значениями, изменять размер массива, выводить все элементы массива, работать как с вещественными и комплексными числами. Для этого были созданы классы в соответствии с заданными условиями и диаграммой.   
Был создан класс “Complex” с собственной реализацией комплексных чисел, и встроен в проект  
В ходе тестирования ожидаемые результаты совпали с результатами работы программы.  
Программа написана согласно положениям и идеям парадигмы ООП.