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 №3**

**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 полиномом на множестве комплексных чисел. Приложение должно включать основной модуль (функция main), модуль «application» и модули «polinom», «array», «complex».   
Разработать класс "Полином" произвольной степени. Класс должен уметь создавать полином 0 степени и n степени (значение задается коэффициентом *an* и N значениями своих корней), хранить корни в массиве, вычислять коэффициенты при степенях х и хранить их в массиве, обеспечивая к ним доступ только на чтение, вводить коэффициент *an* и корни полинома из стандартного потока ввода (объект cin класса istream), выводить элементы массива в стандартный поток вывода (объект cout класса ostream) в форме 1 и 2 по выбору, вычислять значение полинома в заданной точке х.  
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 3. Первичный протокол класса COMPLEX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибуты (старые) | | | | |
| идентификатор | тип | область видимости | | семантическое описание |
| real | double | private | | Действительная часть комплексного числе |
| Imaginary | double | private | | Мнимая часть комплексного числа |
| Методы (старые) | | | | |
| идентификатор | тип | | область видимости | семантическое описание |
| Complex() | - | | public | Конструктор по умолчанию, инициализирует real и imaginary значением 0. |
| Complex(double real, double imaginary) | - | | 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. |

Таблица 4. Первичный протокол класса POLYNOM

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибуты | | | |
| идентификатор | тип | область видимости | семантическое описание |
| roots | std::unique\_ptr<Array> | private | Указатель на массив корней полинома. |
| coefficient | std::unique\_ptr<Array> | private | Указатель на массив коэффициентов полинома. |
| anCoeff | number | private | Ведущий коэффициент полинома. |
| Методы | | | |
| идентификатор | тип | область видимости | семантическое описание |
| Polynom() | - | public | Конструктор по умолчанию, инициализирует массивы корней и коэффициентов. |
| Polynom(number constant) | - | public | Конструктор, принимающий константу для корней и коэффициентов. |
| Polynom(number t\_an, unsigned N) | - | public | Конструктор, инициализирующий массив корней по заданному числу корней. |
| ~Polynom() | - | public | Деструктор для освобождения ресурсов. |
| Create() | bool | public | Создаёт полином, запрашивая корни и коэффициенты у пользователя. |
| Clear() | bool | public | Очищает массивы корней и коэффициентов. |
| Clear() | bool | public | Устанавливает новое значение корня или ведущего коэффициента. |
| multiplyPolynom() | std::vector<number> | public | Умножает полином на корень, возвращая результат. |
| expandPolynomial() | std::vector<number> | public | Преобразует полином из множителей в каноническую форму. |
| evaluateAtPoint() | number | public | Вычисляет значение полинома в заданной точке. |
| PrintCanonicalForm() | void | public | Выводит каноническую форму полинома. |
| PrintNonCanonicalForm() | void | public | Выводит неканоническую форму полинома. |
| GetRoots() | Array\* | public | Возвращает массив корней полинома. |
| GetCoeffs() | Array\* | public | Возвращает массив коэффициентов полинома. |

Диаграмма классов программы:

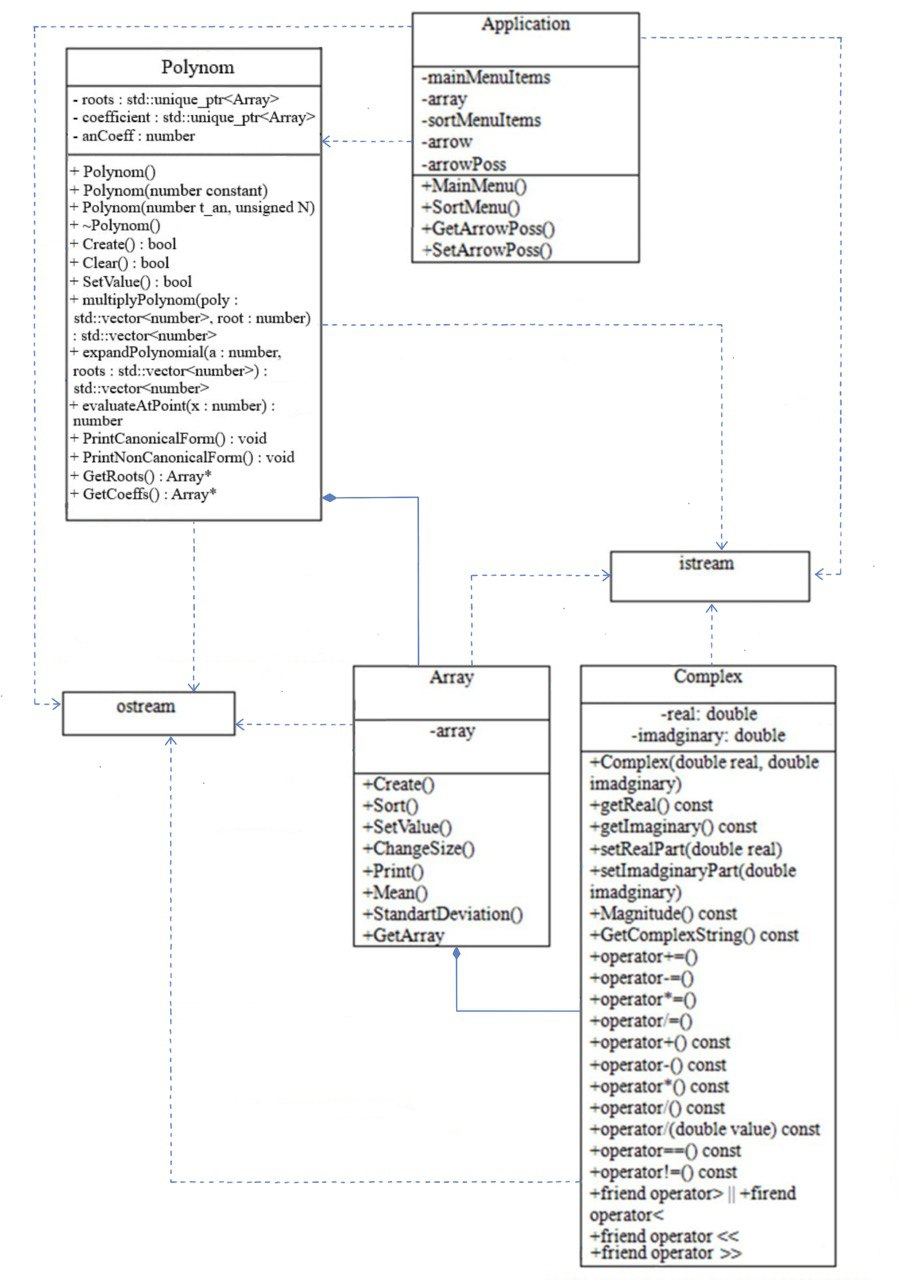


Рис.1. Диаграмма классов работы №3

Тесты работы программы:   
  
Контрольный пример:  
количество корней 3  
An 14+88i

1 корень 1+0i  
2 корень 66+7i

3 корень -15+20i

Точка для вычисления полинома 3-1i

Корень для изменения 10+2i  
  
Исходный полином (14 + 88i) \* (x-(1 + 0i)) \* (x-(66 + 7i)) \* (x-(-15 + 20i))

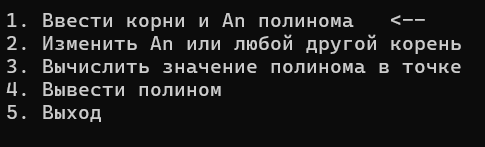
В каноническом виде: (122740+82430i)x³ - (-124402-77564i)x² + (1648-4954i)x + 14 + 88i

Значение в точке 3-1i: -342030-74160i

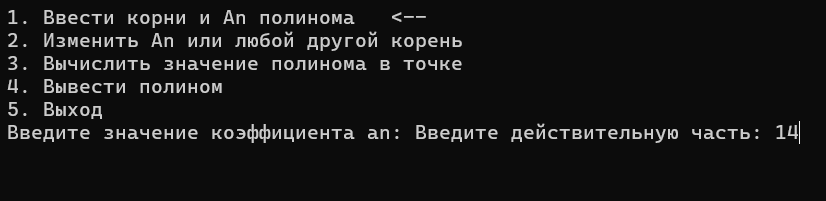
Изменим корень с индексом (по очередности) 1, результатом будет полином: (14 + 88i) \* (x - (1 + 0i)) \* (x - (10+2i)) \* (x - (-15 + 20i))

Проверим программу на описанном контрольном примере:

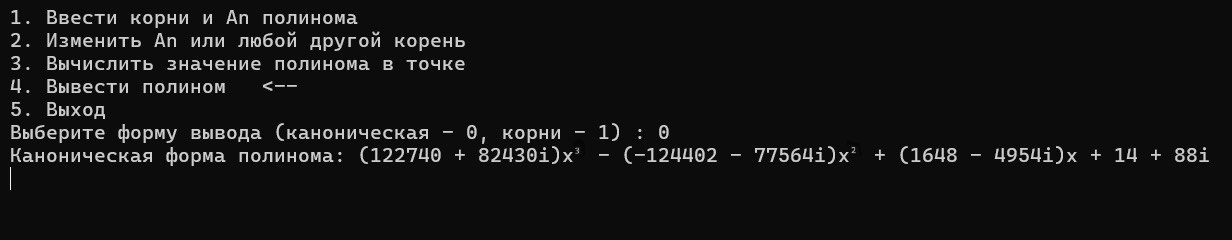
1. Запустим программу



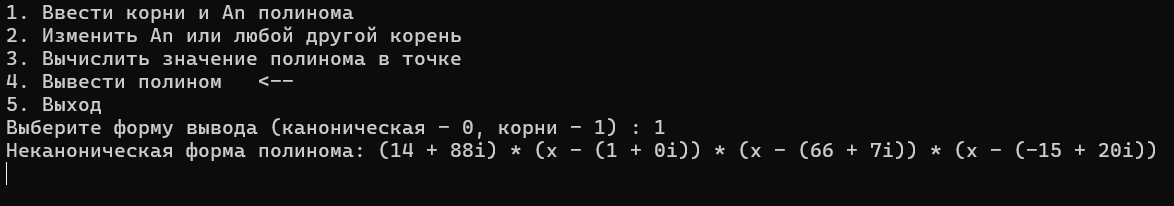
1. Введем значения контрольного примера



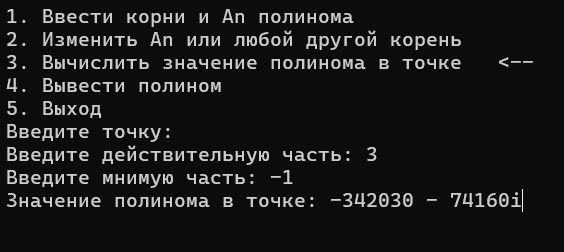
1. Выведем в канонической форме



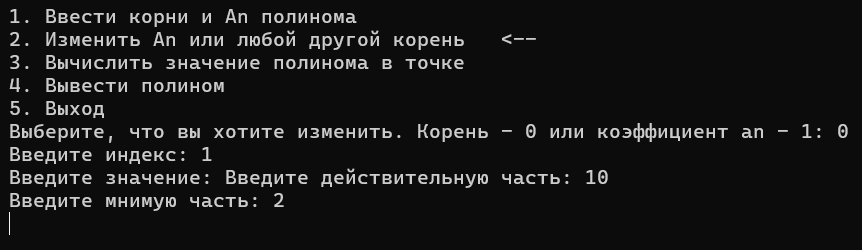
1. В неканонической форме



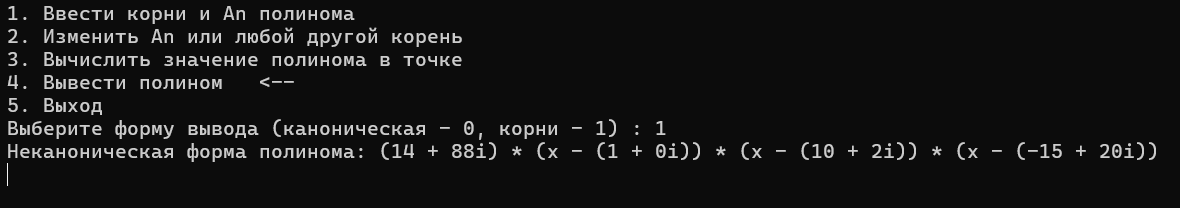
1. Вычислим значение в точке 3-1i



1. Изменим значение корня



1. Выведем еще раз чтобы проверить изменения



Расчетные результаты совпадают с полученными. Программа работает корректно.

**Выводы:**

В ходе работы было создано консольное приложение для работы с полиномами произвольной степени на множестве комплексных чисел, способное: создавать полиномы с коэффициентом *an* и N корнями, вычислять коэффициенты полинома, выводить его в канонической и неканонической формах, изменять коэффициенты и корни, а также вычислять значение полинома в заданной точке.

Для реализации программы были созданы классы в соответствии с заданными условиями и диаграммой классов. В проект был встроен класс "Complex" для работы с комплексными числами.

В ходе тестирования программа показала правильность выполнения всех заявленных функций: ожидаемые результаты совпали с фактическими результатами работы программы. Программа разработана в соответствии с принципами и идеями объектно-ориентированного программирования (ООП).