

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

учреждение образования  
«Гродненский государственный университет имени Янки Купалы»

Факультет математики и информатики  
Кафедра современных технологий программирования

**ЯНЮК ДЕНИС ГЕННАДЬЕВИЧ**  
**Класс complex для работы с комплексными числами**

Курсовая работа  
по дисциплине «Основы алгоритмизации и программирования»  
студента 1 курса специальности  
**1-40 01 01** «Программное обеспечение информационных технологий»  
дневной формы получения образования

Научный руководитель  
Банюкевич Елена Викторовна,  
преподаватель кафедры  
современных технологий  
программирования

Гродно, 2020

## **РЕЗЮМЕ**

**Янюк Денис Геннадьевич**

Курсовая работа – «Класс complex для работы с комплексными числами», 18 страниц, 4 иллюстрации, 5 использованных источников.

Ключевые слова – Комплексные числа, C++, Класс, ООП, Методы класса, Консольное приложение.

Цель исследования – разработать класс complex для работы с комплексными числами.

Объект исследования – класс complex для работы с комплексными числами.

Предмет исследования – реализация класса complex для работы с комплексными числами на ЭВМ с помощью языка C++.

В работе были использованы следующие методы: сравнительный анализ, абстрагирование.

Авторская характеристика работы: класс complex для работы с комплексными числами написан на языке C++ и представляет собой простую реализацию, разработанную программой Microsoft Visual Studio 2019.

## **SUMMARY**

**Yanyuk Denis Gennadievich**

Course work - "complex Class for working with complex numbers", 18 pages, 4 illustrations, 5 sources used.

Keywords-Complex numbers, C++, Class, OOP, class Methods, console application.

The purpose of the research is to develop the complex class for working with complex numbers.

The object of research is the complex class for working with complex numbers.

The subject of research is the implementation of the complex class for working with complex numbers on a computer using the C++language.

The following methods were used: comparative analysis, abstraction.

Author's description of the work: the complex class for working with complex numbers is written in C++ and is a simple implementation developed by Microsoft Visual Studio 2019.

## СОДЕРЖАНИЕ

РЕЗЮМЕ.....	2
ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, УСЛОВНЫХ ОБОЗНАЧЕНИЙ.....	4
ВВЕДЕНИЕ.....	5
ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ.....	6
1.1 Основные аспекты.....	6
1.2 Обзор существующих решений.....	7
1.2.1 Арифметические действия .....	7
1.2.2 Тригонометрическая форма .....	8
1.3 Выводы по главе 1.....	10
ГЛАВА 2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ, СВЯЗАННОЙ С КЛАССОМ COMPLEX .....	11
2.1 Организация интерфейса.....	11
2.2 Работа с комплексными числами .....	11
2.3 Выводы по главе 2.....	12
ГЛАВА 3 РЕАЛИЗАЦИЯ СИСТЕМЫ, СВЯЗАННОЙ С КЛАССОМ COMPLEX.....	13
3.1 Реализация интерфейса.....	13
3.2 Методы, решающие основные задачи .....	15
3.3 Выводы по главе 3.....	16
ЗАКЛЮЧЕНИЕ.....	17
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	18

## ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, УСЛОВНЫХ ОБОЗНАЧЕНИЙ

**ООП**      Объектно-      методология программирования, основанная на  
ориентирован      представлении программы в виде совокупности  
ное      объектов, каждый из которых является  
программиров      экземпляром определённого класса, а классы  
ание      образуют иерархию наследования.

**Класс**      элемент ПО, описывающий абстрактный тип  
данных и его частичную или полную  
реализацию.

**Метод**      функция или процедура, принадлежащая классу  
**Класса**

## ВВЕДЕНИЕ

Решение многих задач физики и техники приводит к квадратным уравнениям с отрицательным дискриминантом. Эти уравнения не имеют решения в области действительных чисел. Но решение многих таких задач имеет вполне определенный физический смысл. Значение величин, получающихся в результате решения указанных уравнений, называли комплексными числами. Комплексные числа широко использовал отец русской авиации Н. Е. Жуковский (1847-1921) при разработке теории крыла, автором которой он является. Комплексные числа и функции от комплексного переменного находят применение во многих вопросах науки и техники.

Цель курсовой работы – разработать класс `complex` для работы с комплексными числами.

Объект исследования – класс `complex` для работы с комплексными числами.

Предмет исследования – реализация класса `complex` для работы с комплексными числами на ЭВМ с помощью языка C++.

В работе были использованы следующие методы: сравнительный анализ, абстрагирование, объектно-ориентированное программирование.

# ГЛАВА 1

## АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

### 1.1 Основные аспекты

Класс — это элемент ПО, описывающий абстрактный тип данных и его частичную или полную реализацию. Другие абстрактные типы данных — метаклассы, интерфейсы, структуры, перечисления, — характеризуются какими-то своими, другими особенностями. Наряду с понятием «объекта» класс является ключевым понятием в ООП (хотя существуют и бесклассовые объектно-ориентированные языки, например, Self, Lua). Суть отличия классов от других абстрактных типов данных состоит в том, что при задании типа данных класс определяет одновременно как интерфейс, так и реализацию для всех своих экземпляров, а вызов метода-конструктора обязателен.

Первоначально идея о необходимости использования комплексных чисел возникла в результате формального решения кубических уравнений, при котором в формуле Кардано под знаком квадратного корня получалось отрицательное число. Большой вклад в исследование комплексных чисел внесли такие математики, как Эйлер, который ввёл общепризнанное обозначение  $i$  для мнимой единицы, Декарт, Гаусс. Сам термин «комплексное число» ввёл в науку Гаусс в 1831 году.

Свойства комплексных чисел и функций нашли широкое применение для решения многих практических задач в различных областях математики, физики и техники: в обработке сигналов, теории управления, электромагнетизме, теории колебаний, теории упругости и многих других. Преобразования комплексной плоскости оказались полезны в картографии и гидродинамике. Современная

физика полагается на описание мира с помощью квантовой механики, которая опирается на систему комплексных чисел.

## 1.2 Обзор существующих решений

Для того, чтобы создать максимально эффективный класс для работы с комплексными числами, нужно изучить материал, который поможет нам установить основные операции, проводимые над комплексными числами.

### 1.2.1 Арифметические действия

Так же, как и для вещественных чисел, для комплексных чисел определены операции сложения, вычитания, умножения и деления. Однако многие свойства комплексных чисел отличаются от свойств вещественных чисел; например, нельзя указать, какое из двух комплексных чисел больше или меньше. Сложение и вычитание происходят по правилу:

$$(a_1 + b_1 i) \pm (a_2 + b_2 i) = (a_1 \pm a_2) + (b_1 \pm b_2) i \quad (1.1)$$

а умножение — по правилу:

$$(a_1 + b_1 i) \cdot (a_2 + b_2 i) = (a_1 \cdot a_2 - b_1 \cdot b_2) + (a_1 \cdot b_2 + a_2 \cdot b_1) i \quad (1.2)$$



Деление чисел осуществляется методом умножения знаменателя и числителя на сопряженное знаменателю выражение. Зная это можно вывести следующую формулу:

$$\frac{(a_1+b_1i)}{(a_1+b_2i)} = \frac{(a_1 \cdot a_2 - b_1 \cdot b_2) + (a_1 \cdot b_2 + a_2 \cdot b_1)i}{a_2^2 + b_2^2} \quad (1.3)$$

### 1.2.2 Тригонометрическая форма

У комплексных чисел есть удобное и наглядное геометрическое представление: число  $z = a + bi$  можно изображать вектором с координатами

$(a; b)$  на декартовой плоскости (или, что почти то же самое, точкой — концом вектора с этими координатами).

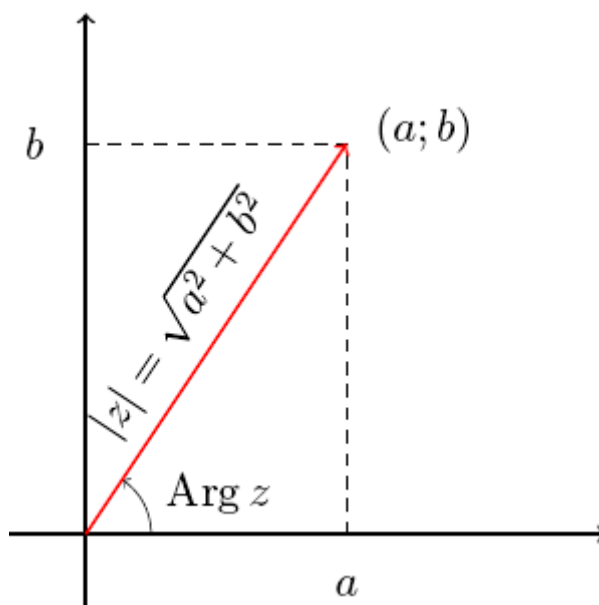


Рисунок 1.

При этом сумма двух комплексных чисел изображается как сумма соответствующих векторов (которую можно найти по правилу

параллелограмма). Длина вектора с координатами (a; b) вычисляется по формуле:

$$|z| = \sqrt{a^2 + b^2} \quad (1.4)$$

Эта величина называется модулем комплексного числа  $z$  и обозначается  $|z|$ . Угол, который этот вектор образует с положительным направлением оси абсцисс (отсчитанный против часовой стрелки), называется аргументом комплексного числа  $z$  и обозначается  $\text{Arg } z$ . Аргумент определен не однозначно, а лишь с точностью до прибавления величины, кратной  $2\pi$  радиан (или  $360^\circ$ , если считать в градусах) — ведь ясно, что поворот на такой угол вокруг начала координат не изменит вектор. Но если вектор длины  $r$  образует угол  $\varphi$  с положительным направлением оси абсцисс, то его координаты равны ( $r \cdot \cos \varphi$ ;  $r \cdot \sin \varphi$ ). Отсюда получается тригонометрическая форма записи комплексного числа:

$$z = |z| \cdot (\cos(\text{Arg } z) + i \sin(\text{Arg } z)) \quad (1.5)$$

Часто бывает удобно записывать комплексные числа именно в такой форме, потому что это сильно упрощает выкладки. Умножение комплексных чисел в тригонометрической форме выглядит очень просто:

$$z_1 \cdot z_2 = |z_1| \cdot |z_2| \cdot (\cos(\text{Arg } z_1 + \text{Arg } z_2) + i \sin(\text{Arg } z_1 + \text{Arg } z_2)) \quad (1.6)$$

При умножении двух комплексных чисел их модули перемножаются, а аргументы складываются. Отсюда следуют формулы Муавра:

$$z^n = |z|^n \cdot (\cos(n \cdot \text{Arg } z) + i \sin(n \cdot \text{Arg } z)) \quad (1.7)$$

С помощью этих формул легко извлекать корни любой степени из комплексных чисел:

$$w = \sqrt[n]{z} = \sqrt[n]{|z|} \cdot \left( \cos\left(\frac{\text{Arg}z + 2\pi k}{n}\right) + i \sin\left(\frac{\text{Arg}z + 2\pi k}{n}\right) \right) \quad (1.8)$$

где  $k$  может принимать любое значение из множества  $\{0, 1, \dots, n - 1\}$ . Это означает, что всегда есть ровно  $n$  корней  $n$ -й степени из комплексного числа (на плоскости они располагаются в вершинах правильного  $n$ -угольника).

### 1.3 Выводы по главе 1

Исходя из всего вышеперечисленного и рассмотренного, можно понять, что для реализации класса `complex` для работы с комплексными числами следует реализовать методы, соответствующие каждой операции с комплексными числами, то есть в нашем классе должны присутствовать минимум шесть методов, включая сложение, вычитание, умножение, деление, возведение в степень и извлечение корня. Но этого может быть недостаточно для полноты картины. Для таких действий, как возведение в степень и извлечение корня лучше использовать тригонометрическую форму комплексного числа, так что следует предусмотреть наличие дополнительных методов класса `complex`, таких как перевод в тригонометрическую форму и показательную форму, а также дополнительные методы, не относящиеся непосредственно к комплексным числам, но не менее важные, организующие работу класса в приложении.

## **ГЛАВА 2**

# **ПРОЕКТИРОВАНИЕ СИСТЕМЫ, СВЯЗАННОЙ С КЛАССОМ COMPLEX**

### **2.1 Организация интерфейса**

Класс зачастую описывает объект реального мира. Как и реальный объект, класс содержит свой набор параметров и характеристик. Каждый такой параметр называется полем класса. Также класс способен манипулировать своими характеристиками (полями) с помощью методов класса.

Для того, чтобы класс раскрыл себя полностью, необходимо организовать взаимодействие класса и пользователя через приложение, для этого нам необходим интерфейс. Для этого я выбрал простой в реализации текстовый интерфейс консольного приложения, так как главная задача – организация работы класса complex. На данном этапе мы можем обозначить основные методы, необходимые для работы интерфейса. Прежде всего должна быть возможность выбора операции, проводимой над комплексным числом, во-вторых возможность ввода комплексного числа, а также проверки введенных данных. И последний метод, не относящийся непосредственно к комплексным числам – вывод данных.

### **2.2 Работа с комплексными числами**

Как нам уже известно, набор операций с комплексными числами не столь велик, поэтому есть возможность реализовать класс complex, содержащий методы, выполняющие все операции, включающие в себя как операции над

одним комплексным числом, так и над двумя комплексными числами сразу. Таким образом мы можем разделить все имеющиеся методы на две части.

Первая часть состоит из методов, работающих с одним комплексным числом. В их число входят:

- Метод перевода комплексного числа в тригонометрическую форму;
- Метод перевода комплексного числа в показательную форму;
- Метод возведения комплексного числа в степень;
- Метод извлечения корня из комплексного числа.

Вторая часть содержит методы, работающие с двумя комплексными числами:

- Метод сложения комплексных чисел;
- Метод вычитания комплексных чисел;
- Метод умножения комплексных чисел;
- Метод деления комплексных чисел.

## **2.3 Выводы по главе 2**

В этой главе были рассмотрены все возможные методы, необходимые для реализации класса `complex`, а также консольного приложения, использующего данный класс.

В итоге все методы были структурированы и подразделены на две части, что упростит и ускорит реализацию класса `complex` для работы с комплексными числами.

## ГЛАВА 3

# РЕАЛИЗАЦИЯ СИСТЕМЫ, СВЯЗАННОЙ С КЛАССОМ COMPLEX

### 3.1 Реализация интерфейса

При проектировании ПО немаловажную роль имеет пользовательский интерфейс приложения. В приложении, использующем класс `complex` для работы с комплексными числами мы используем обычный текстовый интерфейс консольного приложения, предоставленного нам возможностями интегрированной среды разработки Microsoft visual studio.

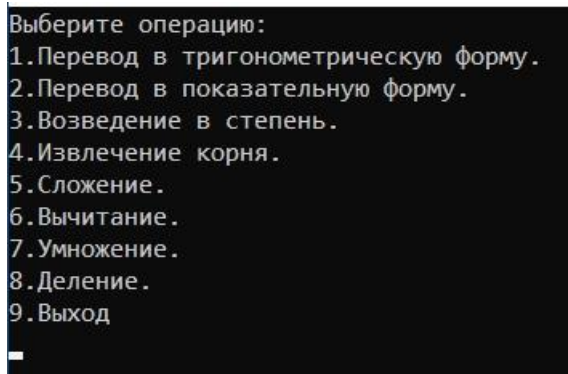


Рисунок 2.

Меню выбора операции было реализовано с помощью условного оператора `if`, а также оператора множественного выбора `switch`. В зависимости от выбора операции пользователем существует вариативность в последующем ходе программы. Так как методы класса уже структурированы и подразделены на 2 типа, существует два варианта развития событий: если пользователь выбирает операции с первой по четвертую включительно, то от него требуется ввод только одного комплексного числа, если же выбраны операции с пятой по восьмую включительно, то необходимо будет ввести два комплексных числа, для соответствующих операций над ними. Выбор девятого варианта просто прервет

работу консольного приложения. Сам ввод реализован в отдельном методе, вызываемом из конструктора класса.

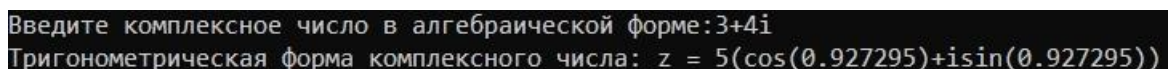
Конструктор класса – это специальный метод, который вызывается при создании нового объекта и используется для инициализации полей класса значениями, а также для начальных вычислений, если они необходимы. После создания объекта конструктор вызвать нельзя. Кроме того, данный метод никогда не возвращает никакого значения.

Поскольку функция ввода вызывается из самого конструктора, передавать никаких значений в конструктор не имеет смысла, таким образом в нашем классе присутствует только один конструктор, объявленный явно, не принимающий никаких значений.

Для того, чтобы можно было инициализировать переменные  $a$  и  $b$ , содержащиеся в классе и отвечающие за действительную и мнимую часть комплексного числа соответственно, передаем в метод переменную  $z$  типа `char*` (строка) и помещаем данные, введенные с клавиатуры, в эту переменную. Чтобы в эту переменную не записывались слова, буквы и прочие элементы, не имеющие отношения к комплексным числам, была реализована посимвольная проверка ввода с помощью ASCII кодов. Сразу же после метода ввода, вызывается метод анализа введенных данных, который в свою очередь анализирует строку  $z$ , рабивает ее на действительную и мнимую части и инициализирует переменные  $a$  и  $b$  с помощью возможности передачи этих переменных в метод по ссылке. Для удобного доступа к любым данным, характеризующим комплексные числа в конструкторе также инициализируется переменная  $r$ , так же известная как  $|z|$ , которая вычисляется по формуле (1.4) и переменная  $argz$ , которая вычисляется более сложным методом, чем переменная  $r$  и имеет отдельный метод для инициализации.

### 3.2 Методы, решающие основные задачи

Мы уже выделили основные операции над комплексными числами и методы, реализующие эти операции. При выборе первой половины операций инициализируется одна переменная сложного типа `complex`. Поскольку все данные инициализируются в конструкторе класса, методы, осуществляющие перевод комплексного числа в тригонометрическую и показательную формы, являются простыми функциями вывода, представляющими данные в нужной форме.



```
Введите комплексное число в алгебраической форме:3+4i
Тригонометрическая форма комплексного числа: z = 5(cos(0.927295)+isin(0.927295))
```

Рисунок 3.

Для тригонометрической формы в виде, представленном формулой (1.5), а для показательной формы в виде:

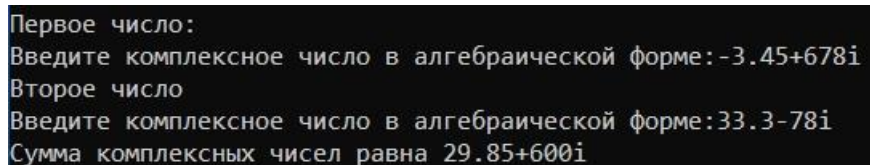
$$z = |z| \cdot e^{i \operatorname{Arg} z} \quad (3.1)$$

В отличие от простых методов перевода в другие формы, методы возведения в степень и извлечения корня более сложны и требуют не только знание формул (1.7), (1.8), но также анализ комплексного числа и разные действия, зависящие от данных, находящихся в переменных `a` и `b`.

После выбора операций, реализующих арифметические действия, происходит объявление двух переменным типа `complex` и их последующая инициализация в конструкторе класса. Функции, выполняющие арифметические действия просты, они содержат формулы (1.1), (1.2), (1.3), а также вызов функции вывода, которая в свою очередь анализирует данные, содержащиеся в



переменных  $a$  и  $b$  и осуществляет вывод числа на консоль в алгебраической форме.



```
Первое число:  
Введите комплексное число в алгебраической форме:-3.45+678i  
Второе число  
Введите комплексное число в алгебраической форме:33.3-78i  
Сумма комплексных чисел равна 29.85+600i
```

Рисунок 4.

### 3.3 Выводы по главе 3

Разработанный класс имеет полный набор методов, взаимодействующих между собой и организовывающих работу с комплексными числами таким образом, что становится возможным проведения любых операций, связанных с комплексными числами. В этой главе были пояснены все аспекты разработки методов, осуществляющих работу класса. В будущем можно разработать приятный графический интерфейс, создав полноценный калькулятор комплексных чисел, который может помочь студентам технических специальностей в освоении такого непростого предмета, как высшая математика.

## ЗАКЛЮЧЕНИЕ

Комплексные числа – один из разделов курса математического анализа более всего подходящий для профессиональной направленности инженеров-бакалавров по направлению подготовки «Информатика и вычислительная техника», а также «Электроэнергетика и электротехника». При изучении комплексных чисел нужно учитывать применение знаний математики в специальных и общетехнических дисциплинах, в частности электротехнике. Использование комплексных чисел дает инженерам возможность пользоваться законами, формулами и методами расчетов, применяющиеся в цепях постоянного тока, для проведения расчетов цепей переменного тока, упрощать различные расчеты, заменив векторно-графическое решение алгебраическими методами, рассчитывать сложные цепи, которые невозможно решить иным путем, упрощать расчеты цепей переменного и постоянного токов.

Они играют значительную роль в таких науках, как физика, химия. В настоящее время комплексные числа активно используются в электромеханике, компьютерной и космической индустрии. Разработка, казалось бы, столь малого элемента в сравнении с космической индустрией, класса для работы с комплексными числами может дать небольшой толчок в понимании студентами комплексных чисел, а также облегчить их изучение.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Информационный источник Wikipedia., <https://ru.wikipedia.org/>
2. Высшая математика для заочников и не только, <http://mathprofi.ru/>
3. Уроки программирования, алгоритмы, статьи, исходники, примеры программ и полезные советы, <https://vscode.ru/>
4. Студенческий научный форум, <https://scienceforum.ru/>
5. Элементы: популярный сайт о фундаментальной науке, <https://elementy.ru/>

# Приложения

## Приложение №1

```
#include <iostream>
#include <Windows.h>
#include <cmath>
#include "complexlib.h"

using namespace std;

const int numberlenght = 10, complexlenght = 2 * numberlenght + 3;
const double pi = 3.14159265358979323846, eps = 0.0001;

int main()
{
    SetConsoleCP(1251);

    SetConsoleOutputCP(1251);
    int menu = 0, number = 0, p = 0;
    for (;;)
    {
        for (; (menu > 9 || menu < 1);)
        {
            cout << "Выберите операцию:\n1.Перевод в тригонометрическую
форму.\n2.Перевод в показательную форму.\n3.Возведение в степень.\n4.Извлечение
корня.\n5.Сложение.\n6.Вычитание.\n7.Умножение.\n8.Деление.\n9.Выход\n";
            cin >> menu;
            cin.ignore(cin.rdbuf()->in_avail());
        }
        if (menu < 5)
        {
            complex z1;
            switch (menu)
            {
            case 1:
            {
                z1.trigform(z1);
                break;
            }
            case 2:
            {
                z1.pokazform(z1);
                break;
            }
            case 3:
            {
                for (int i = 0; p < 1; i++)
                {
                    if (i > 0)
                    {
                        cout << "Неверный ввод\n";
                    }
                    cout << "Введите степень: ";
                    cin >> p;
                    cin.ignore(cin.rdbuf()->in_avail());
                }
            }
            }
        }
    }
}
```

```

    }
    cout << "Комплексное число в " << p << " степени равно";
    z1.power(z1, p);
    break;
}
case 4:
{
    for (int i = 0; p < 1; i++)
    {
        if (i > 0)
        {
            cout << "Неверный ввод\n";
        }
        cout << "Введите степень: ";
        cin >> p;
        cin.ignore(cin.rdbuf()->in_avail());
    }
    cout << "Корень " << p << " степени степени комплексного
числа:\n";

    z1.root(z1, p);
    break;
}
}
else
{
    if (menu < 9)
    {
        cout << "Первое число:\n";
        complex z1;
        cout << "Второе число\n";
        complex z2;
        switch (menu)
        {
            case 5:
            {
                cout << "Сумма комплексных чисел равна ";
                z1.addition(z1, z2);
                break;
            }
            case 6:
            {
                cout << "Разность комплексных чисел равна ";
                z1.substractoin(z1, z2);
                break;
            }
            case 7:
            {
                cout << "Произведение комплексных чисел равно ";
                z1.multiplication(z1, z2);
                break;
            }
            case 8:
            {
                cout << "Частное комплексных чисел равно ";
                z1.division(z1, z2);
                break;
            }
        }
    }
}

```

```

    }
    else
    {
        break;
    }
}
menu = 0;
p = 0;
}
}

```

## Приложение №2

```

#include <iostream>
#include <cmath>
#include "complexlib.h"

using namespace std;

const int numberlength = 10, complexlength = 2 * numberlength + 3;
const double pi = 3.14159265358979323846, eps = 0.0001;

void complex::zinput(char* z)
{
    for (int k = 0; k != 1;)
    {
        cout << "Введите комплексное число в алгебраической форме:";
        cin.getline(z, complexlength);
        cin.ignore(cin.rdbuf()->in_avail());
        int n = strlen(z);
        for (int i = 0, j = 0, ic = 0, signc = 0, numberc = 0; i < n; i++)
        {
            if ((z[i] > 47 && z[i] < 58) || (z[i] == 46))
            {
                numberc++;
                j++;
            }
            else
            {
                if (z[i] == 45 && i == 0)
                {
                    j++;
                }
                else
                {
                    if ((z[i] == 43 || z[i] == 45) && i != 0)
                    {
                        signc++;
                        j++;
                    }
                    else
                    {
                        if (z[i] == 105)
                        {
                            ic++;
                            j++;
                        }
                    }
                }
            }
        }
    }
}

```

```

        }
    }
    if (ic > 1 || signc > 1)
    {
        j--;
    }
    if (i == n && numberc == 0 && z[i] != 105)
    {
        j--;
    }
    if (j == strlen(z))
    {
        k++;
    }
}
if (k == 0)
{
    cout << "Неверный ввод\n";
}
}

void complex::zanalysis(char* z, double& a, double& b)
{
    char* temp = new char[numberlength];
    int n = strlen(z);
    for (int i = 0, j = 0; i < n; i++)
    {
        temp[j] = z[i];
        if ((z[i] < 48 || z[i] > 57) && (z[i] != 46))
        {
            if (z[i] == 45 && j == 0)
            {
                j++;
            }
            else
            {
                if (z[i] == 105)
                {
                    if (i > 0 && z[i - 1] == 45)
                    {
                        b = -1;
                        j = 0;
                    }
                    else
                    {
                        if ((i > 0 && z[i - 1] == 43) || (z[i] == 105 && i
== 0))
                        {
                            b = 1;
                            j = 0;
                        }
                        else
                        {
                            temp[j] = '\\0';
                            b = strtod(temp, NULL);
                            j = 0;
                        }
                    }
                }
            }
        }
    }
}

```

```

        }
        else
        {
            temp[j] = '\0';
            a = strtod(temp, NULL);
            j = 1;
            temp[0] = z[i];
        }
    }
}
else
{
    if (i == n - 1)
    {
        temp[j + 1] = '\0';
        a = strtod(temp, NULL);
    }
    else
    {
        j++;
    }
}
}

void complex::zaddition(double a1, double b1, double a2, double b2)
{
    double a = a1 + a2, b = b1 + b2;
    zoutput(a, b);
}

void complex::zsubtractoin(double a1, double b1, double a2, double b2)
{
    double a = a1 - a2, b = b1 - b2;
    zoutput(a, b);
}

void complex::zmultiplication(double a1, double b1, double a2, double b2)
{
    double a = a1 * a2 - b1 * b2, b = a1 * b2 + a2 * b1;
    zoutput(a, b);
}

void complex::zdivision(double a1, double b1, double a2, double b2)
{
    double zt = a2 * a2 + b2 * b2, a = (a1 * a2 + b1 * b2) / zt, b = (a2 * b1 - a1 * b2) / zt;
    zoutput(a, b);
}

void complex::zoutput(double a, double b)
{
    if (abs(a - 0) < eps)
    {
        if (abs(b - 0) < eps)
        {
            cout << '0';
        }
        else

```



```

        {
            if (abs(b - 1) < eps)
            {
                cout << "i";
            }
            else
            {
                if (abs(b + 1) < eps)
                {
                    cout << "-i";
                }
                else
                {
                    cout << b << "i";
                }
            }
        }
    }
else
{
    cout << a;
    if (b > eps)
    {
        if (abs(b - 1) < eps)
        {
            cout << "+i";
        }
        else
        {
            cout << "+" << b << "i";
        }
    }
    else
    {
        if (b < 0)
        {
            if (b == -1)
            {
                cout << "-i";
            }
            else
            {
                cout << b << "i";
            }
        }
    }
}
cout << endl;
}

void complex::ztrigform(double r, double argz, double a, double b)
{
    if (a == 0 && b == 0)
    {
        cout << "Комплексное число, равное нулю, нельзя перевести в тригонометрическую форму.\n";
    }
    else
    {

```

```

        cout << "Тригонометрическая форма комплексного числа: z = " << r << "(cos(" <<
argz << ") + i sin(" << argz << "))\n";
    }
}

void complex::zpokazform(double, double, double, double)
{
    if (a == 0 && b == 0)
    {
        cout << "Комплексное число, равное нулю, нельзя перевести в показательную
форму.\n";
    }
    else
    {
        cout << "Показательная форма комплексного числа: z = " << r << "e^(" << argz
<< "i)\n";
    }
}

void complex::zpower(double r, double argz, double a1, double b1, int p)
{
    double a, b;
    if (a1 == 0)
    {
        b = pow(b1, p) * pow(-1, p / 2);
        if (p % 2 == 0)
        {
            zoutput(b, 0);
        }
        else
        {
            zoutput(0, b);
        }
    }
    else
    {
        if (b1 == 0)
        {
            a = pow(a1, p);
            zoutput(a, 0);
        }
        else
        {
            a = pow(r, p) * cos(p * argz);
            b = pow(r, p) * sin(p * argz);
            zoutput(a, b);
        }
    }
}

void complex::zroot(double r, double argz, double a1, double b1, int p)
{
    double a = 0, b = 0;
    if (a1 == 0 && b1 == 0)
    {
        cout << "z = 0\n";
    }
    else
    {

```

```

    for (int i = 0; i < p; i++)
    {
        double newarg = (argz + 2 * pi * i) / p, newr = pow(r, 1.0 / p);
        cos(newarg);
        a = newr * cos(newarg);
        b = newr * sin(newarg);
        cout << "z" << i << " = ";
        zoutput(a, b);

        ztrigform(newr, newarg, a, b);
    }
}

double complex::zarg(double a, double b)
{
    if (a > 0)
    {
        if (b == 0)
        {
            return 0;
        }
        else
        {
            return atan(b / a);
        }
    }
    else
    {
        if (a < 0)
        {
            if (b > 0)
            {
                return pi + atan(b / a);
            }
            else
            {
                if (b < 0)
                {
                    return atan(b / a) - pi;
                }
                else
                {
                    return pi;
                }
            }
        }
        else
        {
            if (b > 0)
            {
                return pi / 2;
            }
            else
            {
                if (b < 0)
                {
                    return -pi / 2;
                }
            }
        }
    }
}

```

```

        }
        else
        {
            return 0;
        }
    }
}

complex::complex()
{
    a = 0;
    b = 0;
    z[0] = '0';
    zinput(z);
    zanalysis(z, a, b);
    r = sqrt(a * a + b * b);
    argz = zarg(a, b);
}

void complex::addition(complex z1, complex z2)
{
    zaddition(z1.a, z1.b, z2.a, z2.b);
}

void complex::substractoin(complex z1, complex z2)
{
    zsubstractoin(z1.a, z1.b, z2.a, z2.b);
}

void complex::multiplication(complex z1, complex z2)
{
    zmultiplication(z1.a, z1.b, z2.a, z2.b);
}

void complex::division(complex z1, complex z2)
{
    zdivision(z1.a, z1.b, z2.a, z2.b);
}

void complex::trigform(complex z)
{
    ztrigform(z.r, z.argz, z.a, z.b);
}

void complex::pokazform(complex z)
{
    zpokazform(z.r, z.argz, z.a, z.b);
}

void complex::power(complex z, int p)
{
    zpower(z.r, z.argz, z.a, z.b, p);
}

void complex::root(complex z, int p)
{
    zroot(z.r, z.argz, z.a, z.b, p);
}

```

}

## Приложение № 3

```
#ifndef complexlib_h
#define complexlib_h

class complex
{
private:
    double a, b, r, argz;
    char z[23];
    void zinput(char*);
    void zanalysis(char*, double&, double&);
    void zaddition(double, double, double, double);
    void zsubtractoin(double, double, double, double);
    void zmultiplication(double, double, double, double);
    void zdivision(double, double, double, double);
    void zoutput(double, double);
    void ztrigform(double, double, double, double);
    void zpokazform(double, double, double, double);
    void zpower(double, double, double, double, int);
    void zroot(double, double, double, double, int);
    double zarg(double, double);
public:
    complex();
    void addition(complex, complex);
    void subtractoin(complex, complex);
    void multiplication(complex, complex);
    void division(complex, complex);
    void trigform(complex);
    void pokazform(complex);
    void power(complex, int);
    void root(complex, int);
};

#endif
```