Карагандинский технический университет имени Абылкаса Сагинова

Кафедра: ИВС

**Лабораторная работа №1,2**

По дисциплине: «Технология программирования»

Тема:Разработка программного обеспечения при объектно-ориентированном подходе

**Преподователь:**

В.В.В

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*подпись*) (*дата*)

**Выполнил:**

Ст.гр. ИС-23-2С Лобов М.Ю

Караганда 2024

**Вариант 9.**

Постpоить класс для pаботы с комплексными числами. Класс должен включать соответствующие поля: действительную и мнимую часть числа.

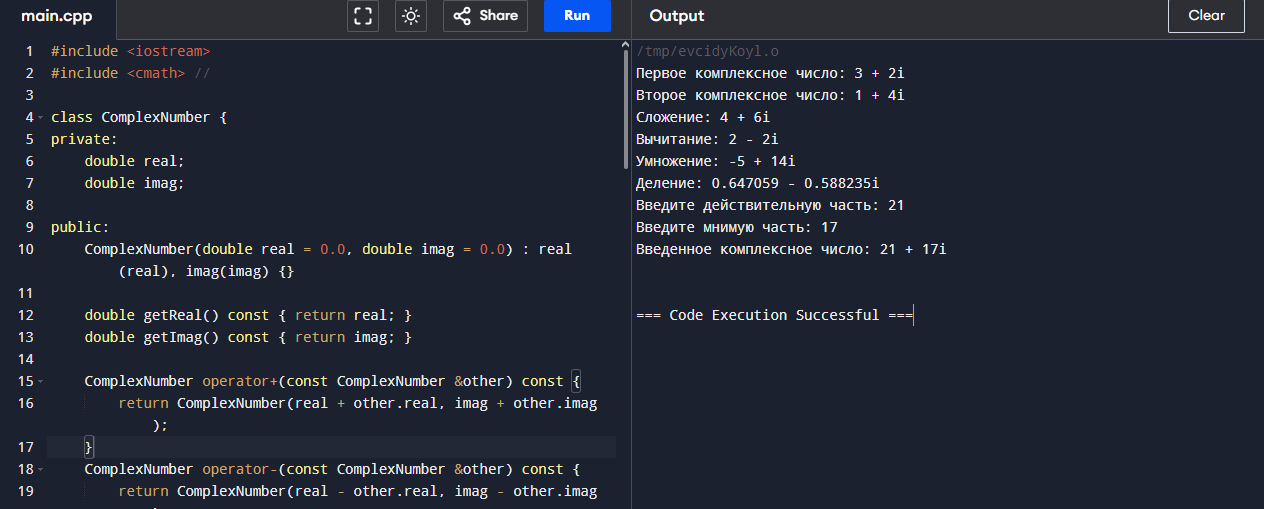
Класс должен обеспечивать пpостейшие функции для pаботы с данными стpуктуpами: сложение, вычитание, умножение, деление, вывод числа в удобной форме и т.д.

Умножение:(a1 + b1i) (a2 + b2i) = (a1a2 - b1b2) + (b1a2 + a1b2)i

Сложение:( a1+b1i) + (a2+b2i) = (a1+a2) + (b1+b2) i.

Вычитание: (a1 + b1i) - (a2 + b2i) = (a1 - a2) + (b1 - b2)i.

Деление: Описание: kompl3_1

 #include <iostream>

#include <cmath> // Для функции sqrt()

class ComplexNumber {

private:

double real; // Действительная часть

double imag; // Мнимая часть

public:

// Конструктор

ComplexNumber(double real = 0.0, double imag = 0.0) : real(real), imag(imag) {}

// Методы для получения действительной и мнимой частей

double getReal() const { return real; }

double getImag() const { return imag; }

// Перегрузка оператора сложения

ComplexNumber operator+(const ComplexNumber &other) const {

return ComplexNumber(real + other.real, imag + other.imag);

}

// Перегрузка оператора вычитания

ComplexNumber operator-(const ComplexNumber &other) const {

return ComplexNumber(real - other.real, imag - other.imag);

}

// Перегрузка оператора умножения

ComplexNumber operator\*(const ComplexNumber &other) const {

double newReal = real \* other.real - imag \* other.imag;

double newImag = imag \* other.real + real \* other.imag;

return ComplexNumber(newReal, newImag);

}

// Перегрузка оператора деления

ComplexNumber operator/(const ComplexNumber &other) const {

double denominator = other.real \* other.real + other.imag \* other.imag;

if (denominator == 0) {

std::cerr << "Ошибка: деление на ноль!" << std::endl;

return ComplexNumber(0, 0);

}

double newReal = (real \* other.real + imag \* other.imag) / denominator;

double newImag = (imag \* other.real - real \* other.imag) / denominator;

return ComplexNumber(newReal, newImag);

}

// Перегрузка оператора вывода

friend std::ostream &operator<<(std::ostream &out, const ComplexNumber &c) {

if (c.imag >= 0)

out << c.real << " + " << c.imag << "i";

else

out << c.real << " - " << -c.imag << "i";

return out;

}

// Перегрузка оператора ввода

friend std::istream &operator>>(std::istream &in, ComplexNumber &c) {

std::cout << "Введите действительную часть: ";

in >> c.real;

std::cout << "Введите мнимую часть: ";

in >> c.imag;

return in;

}

};

// Демонстрационная программа

int main() {

ComplexNumber z1(3, 2); // Создаем первое комплексное число

ComplexNumber z2(1, 4); // Создаем второе комплексное число

std::cout << "Первое комплексное число: " << z1 << std::endl;

std::cout << "Второе комплексное число: " << z2 << std::endl;

// Сложение

ComplexNumber sum = z1 + z2;

std::cout << "Сложение: " << sum << std::endl;

// Вычитание

ComplexNumber difference = z1 - z2;

std::cout << "Вычитание: " << difference << std::endl;

// Умножение

ComplexNumber product = z1 \* z2;

std::cout << "Умножение: " << product << std::endl;

// Деление

ComplexNumber quotient = z1 / z2;

std::cout << "Деление: " << quotient << std::endl;

// Ввод с клавиатуры

ComplexNumber z3;

std::cin >> z3;

std::cout << "Введенное комплексное число: " << z3 << std::endl;

return 0;

}  
***Вопросы к защите лабораторной работы***

1. Что такое класс? Для чего описываются классы?

· **Что такое класс? Для чего описываются классы?** Класс — это шаблон для создания объектов, который определяет свойства (поля) и методы (функции) для работы с этими объектами. Классы описываются для организации и структурирования кода, инкапсуляции данных и создания многоразовых компонентов.

2. Чем отличается класс от структуры?

· **Чем отличается класс от структуры?** Основное отличие заключается в том, что классы обычно используются для создания объектов с более сложным поведением и поддерживают концепцию наследования, тогда как структуры чаще применяются для хранения простых данных. В большинстве языков программирования структуры являются значимыми типами (копируются при передаче), а классы — ссылочными (передаются по ссылке).

3. Какие модификаторы используются для ограничения доступа к членам класса?

**Какие модификаторы используются для ограничения доступа к членам класса?** Основные модификаторы доступа включают:

* public — доступен из любого места.
* private — доступен только внутри самого класса.
* protected — доступен внутри класса и его подклассов.
* internal (в некоторых языках) — доступен только внутри одного сборки.

4. Для чего в классе определяется конструктор? Сколько может быть конструкторов в классе? Когда вызывается конструктор

· **Для чего в классе определяется конструктор? Сколько может быть конструкторов в классе? Когда вызывается конструктор?** Конструктор используется для инициализации объектов класса. В классе может быть несколько конструкторов (перегрузка), каждый из которых может принимать разные параметры. Конструктор вызывается автоматически при создании нового объекта класса.

***Вопросы к защите лабораторной работы***

**1. Какие функции называются дружественными? Как в классе определяются дружественные функции?**

**Дружественные функции** — это функции, которые не являются членами класса, но имеют доступ к его приватным (private) и защищенным (protected) членам. Дружественные функции определяются с помощью ключевого слова friend в классе.

### 2. Отличие в использовании дружественных функций при перегрузке операций от функций-членов.

Функции-члены перегружают операции только для объектов класса, где они определены, и могут работать с одним операндом (левым). Дружественные функции могут перегружать операции для любых объектов и имеют доступ к приватным членам, обеспечивая симметрию операций и возможность работы с двумя операндами (левым и правым).

### 3. Что значит дружественные классы?

**Дружественные классы** — это классы, которые имеют доступ ко всем приватным и защищенным членам другого класса, если они объявлены как friend.

### 4. Как определяются дружественные классы?

Дружественные классы определяются с помощью ключевого слова friend в теле класса.

### 5. Как определяются взаимодружественные классы?

Взаимодружественные классы определяются через использование ключевого слова friend в обоих классах:

### 6. Что такое абстрактный тип данных?

Абстрактный тип данных (АТД) — это математическая модель объекта, в которой определяются только его свойства и поведение (интерфейс), но не реализуется внутренняя структура. В C++ абстрактные типы данных определяются с помощью абстрактных классов (содержат хотя бы одну чисто виртуальную функцию).

### 7. Каковы синтаксис/семантика “операции-функции”?

**Синтаксис операции-функции** соответствует синтаксису обычной функции, но она определяет выполнение операции над объектами класса. Пример:

**Семантика** операции-функции — это логика выполнения операции (например, сложение двух комплексных чисел).

### 8. Как можно вызвать операцию-функцию?

Операцию-функцию можно вызвать как обычную функцию:

Или через оператор:

### 9. Нужно ли перегружать операцию присваивания относительно определенного пользователем типа данных, например класса? Почему?

Да, нужно перегружать оператор присваивания = в случае, если класс управляет динамической памятью или другими ресурсами. Это предотвращает утечку памяти и проблемы с копированием объектов (глубокое копирование).

### 10. Можно ли изменить приоритет перегруженной операции?

Нет, приоритет перегруженной операции нельзя изменить. Приоритет операций определяется синтаксисом языка C++.

### 11. Можно ли изменить количество операндов перегруженной операции?

Нет, количество операндов перегружаемой операции фиксировано для каждого оператора: унарные операторы имеют один операнд, а бинарные — два.

### 12. Можно ли изменить ассоциативность перегруженной операции?

Нет, ассоциативность операторов также фиксирована синтаксисом языка C++ и не может быть изменена.

### 13. Можно ли, используя дружественную функцию, перегрузить оператор присваивания?

Нет, оператор присваивания (=) можно перегрузить только как функцию-член класса.

### 14. Все ли операторы языка C++ могут быть перегружены?

Не все операторы могут быть перегружены. Нельзя перегружать:

* . (оператор доступ к члену класса)
* .\* (указатель на член класса)
* :: (оператор разрешения области видимости)
* ?: (тернарный оператор)
* sizeof (размер объекта)
* typeid
* const\_cast, static\_cast, dynamic\_cast, reinterpret\_cast

### 15. Какими двумя разными способами определяются перегруженные операции?

Перегрузка операций может быть выполнена:

1. Как функция-член класса.
2. Как дружественная или глобальная функция.

### 16. Все ли операции можно перегрузить с помощью глобальной дружественной функции?

Не все. Некоторые операции (например, оператор присваивания) могут быть перегружены только как функции-члены класса.

### 17. В каких случаях операцию можно перегрузить только глобальной функцией?

Глобальные функции нужны для операций, в которых левый операнд не является объектом класса, или для обеспечения симметрии операций, когда оба операнда имеют разные типы.

### 18. В каких случаях глобальная операция-функция должна быть дружественной?

Глобальная операция должна быть дружественной, если ей нужен доступ к приватным членам класса.

### 19. Наследуются ли перегруженные операции?

Да, перегруженные операции наследуются. Однако, при необходимости их можно переопределить в производных классах.

### 20. В чем отличие синтаксиса операции-функции унарной и бинарной операции?

Унарная операция принимает один операнд:

cpp

Копировать код

Complex operator-(); // Унарный минус

Бинарная операция принимает два операнда:

cpp

Копировать код

Complex operator+(const Complex& a);

### 21. Перегрузите операцию “+” для класса “комплексное число”.

cpp

Копировать код

class Complex {

double real, imag;

public:

Complex(double r = 0, double i = 0) : real(r), imag(i) {}

// Перегрузка оператора +

Complex operator+(const Complex& other) const {

return Complex(real + other.real, imag + other.imag);

}

};

### 22. Перегрузите операции “<”, “>”, “==” для класса “строка символов”.

cpp

Копировать код

#include <cstring>

class String {

char\* str;

public:

String(const char\* s = "") {

str = new char[strlen(s) + 1];

strcpy(str, s);

}

// Перегрузка операторов

bool operator<(const String& other) const {

return strcmp(str, other.str) < 0;

}

bool operator>(const String& other) const {

return strcmp(str, other.str) > 0;

}

bool operator==(const String& other) const {

return strcmp(str, other.str) == 0;

}

};

### 23. Что такое поток ввода-вывода?

Поток ввода-вывода — это абстракция, которая позволяет считывать или записывать данные в различные источники и приемники (файлы, консоль и т.д.).

### 24. Что такое файл? В каких форматах сохраняется информация в файлах?

Файл — это хранилище данных на диске. Информация в файлах может храниться в текстовом формате (ASCII) или двоичном формате (побайтовое представление).

### 25. Зачем нужно переопределять операции ввода/вывода >> и << для объектов классов?

Переопределение >> и << упрощает ввод и вывод объектов классов в потоках (например, запись и считывание из файлов или консоли).

### 26. Какие потоки используются при работе с файлами? Где они определены?

Используются потоки ifstream (ввод), ofstream (вывод) и fstream (ввод/вывод). Они определены в заголовочном файле <fstream>.

### 27. С какими атрибутами открывается файл?

Файлы открываются с атрибутами: ios::in (чтение), ios::out (запись), ios::binary (двоичный режим), ios::app (добавление), ios::trunc (усечение).

### 28. Как открыть файл по чтению, по записи, по чтению и записи?

cpp

Копировать код

std::ifstream fileIn("file.txt"); // По чтению

std::ofstream fileOut("file.txt"); // По записи

std::fstream fileInOut("file.txt", std::ios::in | std::ios::out); // По чтению и записи

### 29. Как прочитать информацию из файла?

cpp

Копировать код

std::string line;

std::ifstream file("file.txt");

while (getline(file, line)) {

std::cout << line << std::endl;

}

### 30. Как записать информацию в файл?

cpp

Копировать код

std::ofstream file("file.txt");

file << "Запись в файл" << std::endl;

### 31. Как закрыть файл?

cpp

Копировать код

file.close();

### 32. Как переместить указатель в файле?

Используйте функции seekg() и seekp() для перемещения указателя чтения и записи соответственно:

cpp

Копировать код

file.seekg(10); // Переместить указатель чтения на 10 байт вперед

file.seekp(0, std::ios::end); // Переместить указатель записи в конец файла