Лицей современных технологий управления № 2 г. Пензы

https://pp.userapi.com/c836331/v836331218/6e695/Ivqscmf1s1Y.jpg

**Описание класса по работе с комплексными классами**

Выполнил: ученик 10 класса Нагаев Марат

Руководитель: учитель математики Рожкова Дина Александровна

Пенза 2018

Оглавление

[Введение 3](#_Toc533325731)

[1. Математический аппарат работы с комплексными числами 4](#_Toc533325732)

[2. Описание классов в С++ 7](#_Toc533325733)

[3. Перегрузка операций в С++ 9](#_Toc533325734)

[4. Описание класса по работе с комплексными числами 10](#_Toc533325735)

[5. Визуальное программирование на С++ в Microsoft Visual Studio 17](#_Toc533325736)

[Список используемой литературы 18](#_Toc533325737)

# **Введение**

Объектно-ориентированная парадигма программирования занимает одну из центральных позиций в современном программировании. Анализ предложений работодателей, представленных на различных Интернет-порталах, показал, что более 30% спроса специалистов в области программирования связаны с объектно-ориентированным программированием, с работой в визуальных средах.

Поэтому нашей задачей стало изучение основ ООП на одном из самых востребованных языков программирования – С++.

В качестве содержательного материала были выбраны комплексные числа, математический аппарат показался нам интересным.

Для реализации поставленной цели были определены следующие задачи:

* Рассмотреть основы ООП на языке С++.
* Рассмотреть основные определения и формулы, связанные с работой с комплексными числами.
* Реализовать класс по работе с комплексными числами, который можно рассматривать как своеобразный калькулятор в помощь математику.
* Познакомиться с визуальным программированием на С++ в среде Visual Studio

# **1. Математический аппарат работы с комплексными числами**

Определение 1. Комплексные числа (от лат. complex – совокупный, тесно связанный) – числа вида , где a, b – вещественные числа, i – мнимая единица, то есть число, для которого выполняется равенство . Такая форма записи комплексного числа называется алгебраической.

Существует также тригонометрическая форма записи комплексного числа.

Определение 2. Тригонометрической формой комплексного числа , не равного нулю, называется запись – модуль комплексного числа z,

Термин «комплексное число» ввёл в науку Гаусс в 1831 году. Множество комплексных чисел обычно обозначается символом С, оно содержит множество вещественных чисел R и может рассматриваться как его расширение.

Величина называется вещественной частью числа z. Если , то z называется мнимым или чисто мнимым числом. Вместо обычно пишут просто .

Величина называется мнимой частью числа z. Если , то z является вещественным числом. Вместо обычно пишут просто a. Например, комплексный нуль обозначается просто как 0.

Определим основные операции над комплексными числами.

* Сравнение.

Два комплексных числа и называются равными, если , т.е. равны их действительные и мнимые части.

* Сложения.

Сложение комплексных чисел осуществляется в алгебраической форме и определяется следующим образом: суммой чисел и является число

Т.е. выполняется непосредственное суммирование действительных и мнимых частей.

* Вычитание.

Вычитание комплексных чисел также осуществляется в алгебраической форме. Разность двух чисел чисел и является число

Таким образом, чтобы вычесть из одного числа другое, выполняется непосредственное вычитание действительных и мнимых частей.

* Умножение.

Умножение комплексных чисел в алгебраической форме выполняется непосредственным произведением чисел в алгебраической форме, учитывая свойство мнимой единицы :

.

* Деление.

Частное комплексных чисел в алгебраической форме и находится путем домножения числителя и знаменателя на сопряженное к знаменателю число:

=

* Возведение в степень.

Для возведения комплексного числа в целую степень воспользуемся формулой Муавра.

* (1).

Но в формуле (1) используется тригонометрическая форма комплексного числа. Если же комплексное число представлено в алгебраической форме, то потребуется выполнить преобразование.

Воспользуемся формулами из определения 2: ,

*.*

В результате подстановки в (1) имеем:

(2).

* Извлечение квадратного корня

С помощью формулы Муавра можно получить формулу для расчета корня из комплексного числа

Особо нас заинтересовали поиск синуса и косинуса от комплексного аргумента. Для поиска данных величин необходимо обратиться к показательной функции комплексного аргумента.

Определение 3. Функция, определенная формулой вида для комплексной переменнойназывается экспонентой *z*.

Введем в рассмотрение формулу Эйлера:

Для любого комплексного z выполнено следующее равенство:

(3)

Рассмотрим следующеeе соотношенииe:

(4)

Сложим формулы (3) и (4):

(5)

Аналогично вычтем (3) и (4):

(6)

Формулы (5) и (6) дают определение синуса и косинуса для комплексного аргумента.

Подставим в (5) формулу для экспоненты, представленной в определении 3.

Подставляем в (5):

Далее выполним аналогичную подстановку в (6):

.

# **2. Описание классов в С++**

Класс является абстрактным типом данных, определяемым пользователем, и представляет собой модель реального объекта в виде данных и функций для работы с ними.

Данные класса называются полями (по аналогии с полями структуры), а функции класса – методами. Поля и методы называются элементами класса. Описание класса в первом приближении выглядит так:

class <имя>{

private:

<описание скрытых элементов>

public:

<описание доступных элементов>

};

Спецификаторы доступа private и public управляют видимостью элементов класса. Элементы, описанные после служебного слова private, видимы только внутри класса. Этот вид доступа принят в классе по умолчанию. Интерфейс класса описывается после спецификатора public. Действие любого спецификатора распространяется до следующего спецификатора или до конца класса. Можно задавать несколько секций private и public, порядок их следования значения не имеет.

Конкретные переменные типа «класс» называются экземплярами класса, или объектами.

Доступ к элементам объекта аналогичен доступу к полям структуры. Для этого используются операция . (точка).

Каждый объект содержит свой экземпляр полей класса. Методы класса находятся в памяти в единственном экземпляре и используются всеми объектами совместно, поэтому необходимо обеспечить работу методов с полями именно того объекта, для которого они были вызваны. Это обеспечивается передачей в функцию скрытого параметра this, в котором хранится константный указатель на вызвавший функцию объект. Указатель this неявно используется внутри метода для ссылок на элементы объекта.

Конструктор предназначен для инициализации объекта и вызывается автоматически при его создании. Ниже перечислены основные свойства конструкторов.

* Конструктор не возвращает значение, даже типа void. Нельзя получить указатель на конструктор.
* Класс может иметь несколько конструкторов с разными параметрами для разных видов инициализации (при этом используется механизм перегрузки).
* Конструктор, вызываемый без параметров, называется конструктором по умолчанию.
* Параметры конструктора могут иметь любой тип, кроме этого же класса. Можно задавать значения параметров по умолчанию. Их может содержать только один из конструкторов.
* Если программист не указал ни одного конструктора, компилятор создает его автоматически. Такой конструктор вызывает конструкторы по умолчанию для полей класса и конструкторы по умолчанию базовых классов. В случае, когда класс содержит константы или ссылки, при попытке создания объекта класса будет выдана ошибка, поскольку их необходимо инициализировать конкретными значениями, а конструктор по умолчанию этого делать не умеет.
* Конструкторы не наследуются.
* Конструктор копирования – это специальный вид конструктора, получающий в качестве единственного параметра указатель на объект этого же класса: T::T(const Т&) { /\* Тело конструктора V } где Т ~ имя класса. Этот конструктор вызывается в тех случаях, когда новый объект создается путем копирования существующего

# **3. Перегрузка операций в С++**

C++ позволяет переопределить действие большинства операций так, чтобы при использовании с объектами конкретного класса они выполняли заданные функции. Эта дает возможность использовать собственные типы данных точно так же, как стандартные. Обозначения собственных операций вводить нельзя.

Функция-операция содержит ключевое слово operator, за которым следует знак переопределяемой операции:

тип operator @ ( список параметров) { тело функции }

где @ – символьное обозначение операции.

# **4. Описание класса по работе с комплексными числами**

Перед нами стоят задача описать класс для работы с комплексными числами. В качестве полей класса мы выбрали действительную и мнимую часть комплексного числа, согласно определению 1.

class complex{

double real;

double img;

…

Наш класс будет содержать 3 конструктора: конструктор по умолчанию, конструктор с 2 аргументами и копирующий конструктор.

class complex{

double real;

double img;

public:

complex(){ // Конструктор по умолчанию

real=0; img=0;

}

complex(const complex& c){ // Копирующий конструктор

this->img=c.img;

this->real=c.real;

}

complex(double, double);

…

}

complex::complex(double a, double b){

real=a; img=b;

}

Конструктор определен вне класса

Для комплексного числа в п.1 были определены операции сложения, вычитания, умножения, деления, возведения в степень, вычисление синуса и косинуса. Все данные операции будут реализованы и в классе.

double complex::module(){

return sqrt(real\*real+img\*img);

}

Метод вычисления модуля комплексного числа определен вне класса.

Далее представлено описание методов для вычисления синуса, косинуса и степени комплексного числа, согласно выведенным в п.1 формулам.

complex cosc(){

complex tmp;

tmp.real=cos(this->real)\*(exp(-this->img) + exp(this->img))/2;

tmp.img=sin(this->real)\*(exp(-this->img) - exp(this->img))/2;

return tmp;

}

complex sinc(){

complex tmp;

tmp.real=sin(this->real)\*(exp(-this->img) + exp(this->img))/2;

tmp.img=cos(this->real)\*(-exp(-this->img) + exp(this->img))/2;

return tmp;

}

complex powc(int a){

complex tmp;

tmp.real=pow(this->module(),a)\*cos(a\*this->getf());

tmp.img=pow(this->module(),a)\*sin(a\*this->getf());

return tmp;

}

Для вычисления сложения, вычитания, умножения, деления используем перегрузку операций.

complex operator + (complex c){ // Перегрузка оператора сложения

complex tmp;

tmp.real=this->real+c.real;

tmp.img=this->img+c.img;

return tmp;

}

complex operator - (complex c){

complex tmp;

tmp.real=this->real-c.real;

tmp.img=this->img-c.img;

return tmp;

}

complex operator \* (complex c){

complex tmp;

tmp.real = this->real \* c.real - this->img \* c.img;

tmp.img = this->real \* c.img + this->img \* c.real;

return tmp;

}

complex operator / (complex c)

{

complex tmp;

tmp.real = (this->real \* c.real + this->img \* c.img) / (c.real \* c.real + c.img \* c.img);

tmp.img = (this->img \* c.real - this->real \* c.img) / (c.real \* c.real + c.img \* c.img);

return tmp;

}

Также перегружены операции сравнения, присвоения.

void operator =(complex c){

this->real=c.real;

this->img=c.img;

}

bool operator ==(complex c){

if (this->real==c.real&&this->img==c.img)

return true;

return false;

}

Также мы реализовали метод, который возвращает действительную и мнимую части комплексного числа в виде динамического массива.

double\* returncomplex(){

double\* ret=new double[2];

ret[0]=real;

ret[1]=img;

return ret;

}

Ниже представлено полное описание класса.

#include <iostream>

#include "math.h"

using namespace std;

class complex{

double real;

double img;

public:

complex(){ // Конструктор по умолчанию

real=0; img=0;

}

complex(const complex& c){ // Копирующий конструктор

this->img=c.img;

this->real=c.real;

}

complex(double, double); // Конструктор объявлен в классе, а определен будет вне его

void operator =(complex c){ // Перегрузка оператора присваивания

this->real=c.real;

this->img=c.img;

}

bool operator ==(complex c){ // Перегрузка оператора сравнения

if (this->real==c.real&&this->img==c.img){

return true;}

else {return false;}

}

complex operator + (complex c){ // Перегрузка оператора сложения

complex tmp;

tmp.real=this->real+c.real;

tmp.img=this->img+c.img;

return tmp;

}

complex operator - (complex c){

complex tmp;

tmp.real=this->real-c.real;

tmp.img=this->img-c.img;

return tmp;

}

complex operator \* (complex c){

complex tmp;

tmp.real = this->real \* c.real - this->img \* c.img;

tmp.img = this->real \* c.img + this->img \* c.real;

return tmp;

}

complex operator / (complex c)

{

complex tmp;

tmp.real = (this->real \* c.real + this->img \* c.img) / (c.real \* c.real + c.img \* c.img);

tmp.img = (this->img \* c.real - this->real \* c.img) / (c.real \* c.real + c.img \* c.img);

return tmp;

};

double getf(){

//функция получения угла наклона

double pi=atan(1.00)\*4;

double phi=atan(this->img/this->real);

if (this->real>0) return phi;

if(this->img>0) return phi+pi;

if(this->img<0) return phi-pi;

if (this->img==0&&this->real==0) return 0;

if(this->img==0&&this->real<0) return pi;

}

complex cosc(){

complex tmp;

tmp.real=cos(this->real)\*(exp(-this->img) + exp(this->img))/2;

tmp.img=sin(this->real)\*(exp(-this->img) - exp(this->img))/2;

return tmp;

}

complex sinc(){

complex tmp;

tmp.real=sin(this->real)\*(exp(-this->img) + exp(this->img))/2;

tmp.img=cos(this->real)\*(-exp(-this->img) + exp(this->img))/2;

return tmp;

}

complex powc(int a){

complex tmp;

tmp.real=pow(this->module(),a)\*cos(a\*this->getf());

tmp.img=pow(this->module(),a)\*sin(a\*atan(this->getf());

return tmp;

}

double module(); // Модуль комплексного числа, в классе только прототип

double\* returncomplex(){

double\* ret=new double[2];

ret[0]=real;

ret[1]=img;

return ret;

}

};

complex::complex(double a, double b){ // Конструктор определен будет вне класса

real=a; img=b;

}

double complex::module(){ // Метод определен вне класса

return sqrt(real\*real+img\*img);

}

# **5. Визуальное программирование на С++ в Microsoft Visual Studio**

Для создания графического интерфейса пользователя была выбрана среда от Microsoft как одна из самых популярных и мощных сред для Windows.

В процессе создания программы мы использовали следующие компоненты Windows Forms:

Form

Menu

Button

TextBox

Panel

На скриншоте рис.1 представлен результат работы программы.

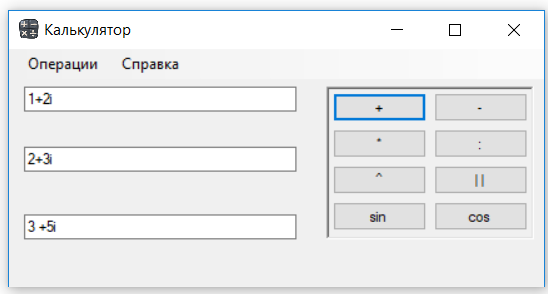


Рис.1. Скриншот результата работы программы

# **Список используемой литературы**

1. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб.: Питер, 2003. – 461 с: