

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННО БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Учебный Центр Информационных Технологий «Информатика»



Лабораторная работа № 1  
по дисциплине «Объектно-ориентированное программирование»

Выполнил слушатель: Пешков Е.В.  
Вариант: 9  
Дата сдачи:  
Преподаватель: Васюткина И.А.

Новосибирск, 2019г.

**Задание:** Построить класс для работы с комплексными числами. Класс должен включать соответствующие поля: действительную и мнимую часть числа, строковое представление комплексного числа. Класс должен обеспечивать простейшие функции для работы с данными класса: сложение, вычитание, умножение, деление, вывод числа в удобной форме на экран и т.д.

## 1. Постановка задачи.

В соответствии с заданием необходимо построить класс для работы с комплексными числами. Класс должен включать следующее:

Поля класса:

- `double re` – действительная часть числа,
- `double im` – мнимая часть числа,
- `char* number` – строковое представление числа,
- `static int count` – статическое поле класса для хранения количества созданных объектов класса

Конструкторы:

- `Complex(double _re, double _im)` – конструктор с параметрами
- `Complex()` – конструктор без параметров
- `Complex(Complex& object)` – конструктор копирования

Деструктор:

- `~Complex ()` – деструктор объектов класса без параметров

Методы класса:

- `void addition(const Complex&, const Complex&)` – сложение комплексных чисел
- `void subtraction(const Complex&, const Complex&)` – вычитание комплексных чисел
- `void multiplication(const Complex&, const Complex&)` – умножение комплексных чисел
- `void division(const Complex&, const Complex&)` – деление комплексных чисел
- `void print()` - вывод числа в консоль в алгебраической форме записи
- `char* toSring(double, double)` – преобразование комплексного числа в строку для записи в поле `number`
- `static void Numobject()` – вывод количества созданных объектов в программе

## 2. Определение пользовательского класса.

### Шаблон класса

```
#pragma warning(disable : 4996)
#include <iostream>
#include <string>
#include <locale>

class Complex
{
    double re;
    double im;
    char* number;
    static int count;

public:
    // Конструкторы
    Complex(double _re, double _im);
    Complex();
    Complex(const Complex &object);

    // Деструктор
    ~Complex();

    // Методы
    void addition(const Complex&);
    void subtraction(const Complex&);
    void multiplication(const Complex&);
    void division(const Complex&);
    void print();
    char* toSring(double, double);
    static void Numobject();
};
```

#### // Конструктор с параметрами

```
Complex::Complex(double _re = 0, double _im = 0)
```

*Параметры:*

double \_re – действительная часть числа.

double \_im – мнимая часть числа.

*Возвращаемый результат:* ничего не возвращает.

*Принцип работы:* создает объект класса Complex.

#### // Конструктор без параметров

```
Complex::Complex()
```

*Параметры:*

*Возвращаемый результат:* ничего не возвращает.

*Принцип работы:* создает объект класса Complex.

#### // Конструктор копирования

```
Complex::Complex(const Complex& object)
```

*Параметры:*

object – объект класса Complex

*Возвращаемый результат:* ничего не возвращает.

*Принцип работы:* выполняет копирование одного объекта класса Complex в другой.

**// Деструктор**

Complex:: ~Complex( )

*Параметры:*

*Возвращаемый результат:* ничего не возвращает.

*Принцип работы:* удаляет объект класса Complex.

**// Метод сложения комплексных чисел**

void Complex::addition ( const Complex& a2)

*Параметры:*

const Complex& a2– ссылка на второе число.

*Возвращаемый результат:* ничего не возвращает.

*Принцип работы:* выполняет сложение комплексных чисел.

**// Метод вычитания комплексных чисел**

void Complex:: subtraction ( const Complex& a2)

*Параметры:*

const Complex& a2– ссылка на второе число.

*Возвращаемый результат:* ничего не возвращает.

*Принцип работы:* выполняет вычитание второго числа из первого комплексного числа.

**// Метод перемножения комплексных чисел**

void Complex:: multiplication (const Complex& a2)

*Параметры:*

const Complex& a2– ссылка на второе число.

*Возвращаемый результат:* ничего не возвращает.

*Принцип работы:* выполняет перемножение первого числа на второе число.

**// Метод деления комплексных чисел**

void Complex:: division (const Complex& a2)

*Параметры:*

const Complex& a2– ссылка на второе число.

*Возвращаемый результат:* ничего не возвращает.

*Принцип работы:* выполняет деление первого числа на второе число.

**// Метод вывода числа в консоль в алгебраической форме**

void Complex::print ( )

*Параметры:*

*Возвращаемый результат:* ничего не возвращает.

*Принцип работы:* выводит в консоль комплексное число в алгебраической форме записи.

**// Метод преобразования комплексного числа в строку**

char\* Complex::toString(double \_re, double \_im)

*Параметры:*

double \_re – действительная часть числа.

double \_im – мнимая часть числа.

*Возвращаемый результат:* возвращает указатель на строку.

*Принцип работы:* преобразование комплексного числа в строку для записи в поле number.

**// Метод подсчёта количества созданных объектов**

void Complex::Numobject()

*Параметры:*

*Возвращаемый результат:* ничего не возвращает.

*Принцип работы:* выводит в консоль количество созданных объектов класса Complex.

### 3. Листинг тестовой программы.

```
#include "Complex.h"

int main()
{
    setlocale(LC_ALL, "Russian");
    std::cout << "Количество объектов = ";
    Complex::Numobject();
    Complex a1(10.222, 6.2);
    Complex a2(5.1, 3.111);
    Complex a3 = a1;
    std::cout << "Количество объектов = ";
    Complex::Numobject();
    a1.print();
    a2.print();
    a3.print();

    Complex x;
    std::cout << "Количество объектов = ";
    Complex::Numobject();

    a1.addition(a2);
    std::cout << "Сложение ";
    a1.print();
    std::cout << "Количество объектов = ";
    Complex::Numobject();

    a1.subtraction(a2);
    std::cout << "Вычитание ";
    a1.print();
    std::cout << "Количество объектов = ";
    Complex::Numobject();

    a1.multiplication(a2);
    std::cout << "Умножение ";
    a1.print();
    std::cout << "Количество объектов = ";
    Complex::Numobject();

    a1.division(a2);
    std::cout << "Деление ";
    a1.print();
    std::cout << "Количество объектов = ";
    Complex::Numobject();

    Complex* pm = new Complex(15.00, 10.00);
    pm->print();
    Complex::Numobject();
    delete pm;

    Complex* parr = new Complex[5];
    Complex::Numobject();
    for (int i = 0; i < 5; i++)
    {
        parr[i].print();
    }
    delete[] parr;
    Complex::Numobject();

    _getwch();
    return 0;
}
```

#### 4. Пример работы программы.

```
E:\NSTU\2019\OOP\NSTU_OOP\Debug\Lab_1_v-9.exe
Количество объектов = 0
Конструктор с параметрами 001AFE74
Конструктор с параметрами 001AFE54
Конструктор копирования 001AFE34
Количество объектов = 3
( 10,222 + 6,2*i )

( 5,1 + 3,111*i )

( 10,222 + 6,2*i )

Конструктор без параметров 001AFE14
Количество объектов = 4
Сложение ( 15,322 + 9,311*i )

Количество объектов = 4
Вычитание ( 10,222 + 6,2*i )

Количество объектов = 4
Умножение ( 32,844 + 133,8*i )

Количество объектов = 4
Деление ( 16,357 + 17,694*i )

Количество объектов = 4
Конструктор с параметрами 0043EC60
( 15, + 10,*i )

5
Вызов деструктора 0043EC60
Конструктор без параметров 0043F62C
Конструктор без параметров 0043F644
Конструктор без параметров 0043F65C
Конструктор без параметров 0043F674
Конструктор без параметров 0043F68C
9
( 0, + 0,*i )

( 0, + 0,*i )

( 0, + 0,*i )

( 0, + 0,*i )

( 0, + 0,*i )

Вызов деструктора 0043F68C
Вызов деструктора 0043F674
Вызов деструктора 0043F65C
Вызов деструктора 0043F644
Вызов деструктора 0043F62C
4
```

## Приложение. Текст программы

### Файл main.c

```
#include "Complex.h"

int main()
{
    setlocale(LC_ALL, "Russian");
    std::cout << "Количество объектов = ";
    Complex::Numobject();
    Complex a1(10.222, 6.2);
    Complex a2(5.1, 3.111);
    Complex a3 = a1;
    std::cout << "Количество объектов = ";
    Complex::Numobject();
    a1.print();
    a2.print();
    a3.print();

    Complex x;
    std::cout << "Количество объектов = ";
    Complex::Numobject();

    x.addition(a1, a2);
    std::cout << "Сложение ";
    x.print();
    std::cout << "Количество объектов = ";
    Complex::Numobject();

    x.subtraction(a1, a2);
    std::cout << "Вычитание ";
    x.print();
    std::cout << "Количество объектов = ";
    Complex::Numobject();

    x.multiplication(a1, a2);
    std::cout << "Умножение ";
    x.print();
    std::cout << "Количество объектов = ";
    Complex::Numobject();

    x.division(a1, a2);
    std::cout << "Деление ";
    x.print();
    std::cout << "Количество объектов = ";
    Complex::Numobject();
    Complex* pm = new Complex(15.00, 10.00);
    pm->print();
    Complex::Numobject();
    delete pm
    Complex* parr = new Complex[5];
    Complex::Numobject();
    for (int i = 0; i < 5; i++)
    {
        parr[i].print();
    }
    delete[]parr;
    Complex::Numobject();

    _getwch();
    return 0;
}
```



## Файл Complex.h

```
#pragma warning(disable : 4996)
#include <iostream>
#include <string>
#include <locale>

class Complex
{
    double re;
    double im;
    char* number;
    static int count;

public:
    // Конструкторы
    Complex(double _re, double _im);
    Complex();
    Complex(const Complex &object);

    // Деструктор
    ~Complex();

    // Методы
    void addition(const Complex&);
    void subtraction(const Complex&);
    void multiplication(const Complex&);
    void division(const Complex&);
    void print();
    char* toSring(double, double);
    static void Numobject();
};
```

## Файл Complex.cpp

```
#include "Complex.h"

// Конструктор с параметрами
Complex::Complex(double _re = 0, double _im = 0) {
    std::cout << "Конструктор с параметрами " << this << std::endl;
    re = _re;
    im = _im;
    char* _number = toSring(re, im);
    number = new char[strlen(_number) + 1];
    strcpy(number, _number);
    delete[] _number;
    count++;
}

// Конструктор без параметров
Complex::Complex() {
    std::cout << "Конструктор без параметров " << this << std::endl;
    re = im = 0.0;
    char* _number = toSring(re, im);
    number = new char[strlen(_number) + 1];
    strcpy(number, _number);
    delete[] _number;
    count++;
}

//Конструктор копирования
Complex::Complex(const Complex& object)
{
    std::cout << "Конструктор копирования " << this << std::endl;
    re = object.re;
    im = object.im;
    number = _strdup(object.number);
    count++;
}
```

```

}
;
// Деструктор
Complex::~Complex()
{
    std::cout << "Вызов деструктора " << this << std::endl;
    if (number)
        delete[] number;
    number = nullptr;
    count--;
}

// Методы
void Complex::addition(const Complex& a2)
{
    re = re + a2.re;
    im = im + a2.im;
    number = toString(re, im);
}

void Complex::subtraction(const Complex& a2)
{
    re = re - a2.re;
    im = im - a2.im;
    number = toString(re, im);
}

void Complex::multiplication(const Complex& a2)
{
    re = (re * a2.re - im * a2.im);
    im = (re * a2.im + a2.re * im);
    number = toString(re, im);
}

void Complex::division(const Complex& a2)
{
    re = (re * a2.re + im * a2.im) / ((a2.re * a2.re) + (a2.im * a2.im));
    im = (a2.re * im - re * a2.im) / ((a2.re * a2.re) + (a2.im * a2.im));
    number = toString(re, im);
}

void Complex::print()
{
    std::cout << "( " << number << " )\n\n";
}

char* Complex::toString(double _re, double _im)
{
    char* _number = new char[50];
    char buf[20];
    _gcvt_s(buf, _re, 5);
    strcpy(_number, buf);
    _number = strcat(_number, " + ");
    _gcvt_s(buf, _im, 5);
    _number = strcat(_number, buf);
    _number = strcat(_number, "*i");
    return _number;
}

int Complex::count = 0;

void Complex::Numobject()
{
    std::cout << count << std::endl;
}

```