*I. Теоретическая часть*

-

*II. Чтение кода*

A B

A\_copy B\_copy

A\_copy B\_move

0

*III. Написание кода*

template <typename T>

class Complex {

private:

T real;

T imag;

public:

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

T getReal() const { return real; }

T getImag() const { return imag; }

Complex operator+(const Complex& other) const {

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

}

Complex operator-(const Complex& other) const {

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

}

Complex& operator+=(const Complex& other) {

real += other.real;

imag += other.imag;

return \*this;

}

Complex& operator-=(const Complex& other) {

real -= other.real;

imag -= other.imag;

return \*this;

}

Complex operator-() const {

return Complex(-real, -imag);

}

Complex& operator=(const Complex& other) {

if (this != &other) {

real = other.real;

imag = other.imag;

}

return \*this;

}

friend std::ostream& operator<<(std::ostream& os, const Complex& c) {

os << c.real << (c.imag >= 0 ? "+" : "") << c.imag << "i";

return os;

}

};