```
Script started on 2024-09-15 13:15:27-05:00 [TERM="xterm-256color" TTY="/dev/pts/3'
e_prevost@ares:~/Portfolio_1/Lab 3$ pwd
/home/students/e_prevost/Portfolio_1/Lab 3
e_prevost@ares:~/Portfolio_1/Lab 3$ cat compclass.info
****************

Robert Prevost

CSC 122 W01



Complex Class Lab



Allows for class type of a Complex number. Complex number contains a real

and imaginary variable. This class allows us to create this object and do

arithmetic on it as well.



Base Level: Level 3

Total Level: Level 3

****************e_prevost@ares:~/Portfolio_1/Lab 3$ show-code compclass.cpp


compclass.cpp:
```

```cpp
#include <iostream>
#include <cmath>

class Complex {
private:
    double real;
    double imag;

public:
    // Constructors
    Complex() : real(0), imag(0) {}
    Complex(double r, double i) : real(r), imag(i) {}

    // Accessors
    double getReal() const { return real; }
    double getImag() const { return imag; }

    // Mutators
    void setReal(double r) { real = r; }
    void setImag(double i) { imag = i; }

    // Basic operations
    Complex add(const Complex& other) const {
        return Complex(real + other.real, imag + other.imag);
    }

    Complex subtract(const Complex& other) const {
        return Complex(real - other.real, imag - other.imag);
    }

    Complex multiply(const Complex& other) const {
        return Complex(real * other.real - imag * other.imag,
                       real * other.imag + imag * other.real);
    }

    Complex divide(const Complex& other) const {
        double denominator = other.real * other.real + other.imag * other.
        imag;
        return Complex((real * other.real + imag * other.imag) /
        denominator,
                       (imag * other.real - real * other.imag) /
                       denominator);
    }

    Complex negate() const {
        return Complex(-real, -imag);
    }

    double magnitude() const {
        return std::sqrt(real * real + imag * imag);
    }

    Complex conjugate() const {
        return Complex(real, -imag);
    }

    // Input/Output
    void input() {
        std::cin >> real >> imag;
    }

    void output() const {
        std::cout << real << " + " << imag << "i";
    }
};

// Driver program
int main() {
    Complex c1(3, 4);
    Complex c2(1, 2);

    std::cout << "c1: ";
    c1.output();
    std::cout << std::endl;

    std::cout << "c2: ";
    c2.output();
    std::cout << std::endl;

    Complex sum = c1.add(c2);
    std::cout << "Sum: ";
    sum.output();
    std::cout << std::endl;

    Complex product = c1.multiply(c2);
    std::cout << "Product: ";
    product.output();
    std::cout << std::endl;

    std::cout << "Magnitude of c1: " << c1.magnitude() << std::endl;

    Complex conj = c1.conjugate();
    std::cout << "Conjugate of c1: ";
    conj.output();
    std::cout << std::endl;

    return 0;
}
e_prevost@ares:~/Portfolio_1/Lab 3$ CPP compclass
compclass.cpp***


e_prevost@ares:~/Portfolio_1/Lab 3$ ./compclass.out
c1: 3 + 4i
c2: 1 + 2i
Sum: 4 + 6i
Product: -5 + 10i
Magnitude of c1: 5
Conjugate of c1: 3 + -4i
e_prevost@ares:~/Portfolio_1/Lab 3$ cat compclass.tpq
```

1. Why do your class methods take fewer arguments than you would expect?

The arguments do not take much of anything because the real and imaginary number values are embedded into the complex number object

2. Does the compiler change y when you have 'x + y' in your program? So should your addition method change the other Complex number (the argument object)? How can you tell the compiler this in the most efficient way?

No, the compiler does not change the value of the number. The simplest way to tell the compiler to not change the argument object is to create a new object to return rather than outputting the object.

3. Does this phenomenon extend to the other operations?

Yes it does. We need to make sure we aren't directly modifying for the other operations.

4. What kind of value should be returned from the standard math operations (i.e. what TYPE of value)? From conjugate? From magnitude?

The result of the standard math operations will always be another complex number so it is best for us to just return a complex number object.

5. Does your input method prompt the user? Why should it not?

No, the input method does not prompt the user. This allows for more versatility with the program to be used across the board instead of forcing the prompt to be a certain way.

6. Does your output method print anything besides the Complex number (even an endl)? Why should it not?

The output method should only output the complex number. Adding an endl would decrease the output's flexibility to be used in programs. Some programs might not want that endl and therefore it is better practice to

```
only output the bare bones of what is needed.e_prevost@ares:~/Portfolio_1/Lab 3$ ex
exit

Script done on 2024-09-15 13:16:09-05:00 [COMMAND_EXIT_CODE="0"]
```