You can use various programming languages **if you’re not using Ada for programming complex numbers and the four basic operations** (addition, subtraction, multiplication, and division).

**You can choose any of these languages based on your familiarity and requirements!**

Here’s a quick overview of how you might implement complex numbers in a few different languages:

Python:

class Complex:

def \_\_init\_\_(self, real, imag):

self.real = real

self.imag = imag

def \_\_add\_\_(self, other):

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

def \_\_sub\_\_(self, other):

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

def \_\_mul\_\_(self, other):

return Complex(self.real \* other.real - self.imag \* other.imag,

self.real \* other.imag + self.imag \* other.real)

def \_\_truediv\_\_(self, other):

denom = other.real\*\*2 + other.imag\*\*2

return Complex((self.real \* other.real + self.imag \* other.imag) / denom,

(self.imag \* other.real - self.real \* other.imag) / denom)

def \_\_str\_\_(self):

return f"{self.real} + {self.imag}i"

# Example usage

a = Complex(2, 3)

b = Complex(4, 5)

print(a + b) # Output: 6 + 8i

print(a \* b) # Output: -7 + 22i

C++

#include <iostream>

class Complex {

public:

double real, imag;

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

Complex operator+(const Complex& other) {

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

}

Complex operator-(const Complex& other) {

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

}

Complex operator\*(const Complex& other) {

return Complex(real \* other.real - imag \* other.imag,

real \* other.imag + imag \* other.real);

}

Complex operator/(const Complex& other) {

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

return Complex((real \* other.real + imag \* other.imag) / denom,

(imag \* other.real - real \* other.imag) / denom);

}

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

return os << c.real << " + " << c.imag << "i";

}

};

// Example usage

int main() {

Complex a(2, 3);

Complex b(4, 5);

std::cout << (a + b) << std::endl; // Output: 6 + 8i

std::cout << (a \* b) << std::endl; // Output: -7 + 22i

return 0;

}

Java

public class Complex {

private double real;

private double imag;

public Complex(double r, double i) {

this.real = r;

this.imag = i;

}

public Complex add(Complex other) {

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

}

public Complex subtract(Complex other) {

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

}

public Complex multiply(Complex other) {

return new Complex(this.real \* other.real - this.imag \* other.imag,

this.real \* other.imag + this.imag \* other.real);

}

public Complex divide(Complex other) {

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

return new Complex((this.real \* other.real + this.imag \* other.imag) / denom,

(this.imag \* other.real - this.real \* other.imag) / denom);

}

@Override

public String toString() {

return real + " + " + imag + "i";

}

// Example usage

public static void main(String[] args) {

Complex a = new Complex(2, 3);

Complex b = new Complex(4, 5);

System.out.println(a.add(b)); // Output: 6.0 + 8.0i

System.out.println(a.multiply(b)); // Output: -7.0 + 22.0i

}

}