Порівняння реалізацій комплексних чисел на мові C та C++

Кухарук Соломія, комп'ютерна математика 1

6.12.24

Анотація

У даному рефераті розглядаються три реалізації роботи з комплексними числами: власноруч написані бібліотеки на мовах C та C++, а також стандартні бібліотеки на кожній з цих мов. Проводиться детальний опис кожної з реалізацій з прикладами коду, а також порівняння швидкості виконання функцій, що працюють з комплексними числами. Для порівняння використовується файл `cross_tests.cpp`, що дозволяє виміряти час роботи бібліотек.

1 Вступ

Комплексні числа є важливою частиною математичних обчислень і часто використовуються в різних наукових та інженерних задачах. У цьому рефераті розглядаються три реалізації бібліотек для роботи з комплексними числами: власноруч написані реалізації на мовах C та C++, а також стандартні бібліотеки, які надають ці мови для роботи з комплексними числами.

2 Реалізація бібліотеки на C

2.1 Функції для реалізації на C

Ось список функцій, реалізованих у бібліотеці на C:

- **Complex add(Complex a, Complex b)**: Функція для додавання двох комплексних чисел.
- **Complex subtract(Complex a, Complex b)**: Функція для віднімання двох комплексних чисел.

- **Complex multiply(Complex a, Complex b)**: Функція для множення двох комплексних чисел.
- **Complex divide(Complex a, Complex b)**: Функція для ділення одного комплексного числа на інше.
- **double modulus(Complex a)**: Функція для обчислення модуля (модульний розмір) комплексного числа.
- **void printComplex(Complex a)**: Функція для виведення комплексного числа на екран.
- **Complex readComplex()**: Функція для введення комплексного числа з клавіатури.
- **void saveToFile(FILE *file, Complex a)**: Функція для збереження комплексного числа у файл.
- **Complex readFromFile(FILE *file)**: Функція для зчитування комплексного числа з файлу.

3 Реалізація бібліотеки на C++

3.1 Функції для реалізації на C++

Ось список функцій, реалізованих у бібліотеці на C++:

- **Complex::Complex()** та **Complex::Complex(double real, double imag)**: Конструктори класу для ініціалізації комплексного числа.
- **Complex Complex::operator+(const Complex other) const**: Перевантаження оператора додавання для комплексних чисел.
- **Complex Complex::operator-(const Complex other) const**: Перевантаження оператора віднімання для комплексних чисел.
- **Complex Complex::operator*(const Complex other) const**: Перевантаження оператора множення для комплексних чисел.
- **Complex Complex::operator/(const Complex other) const**: Перевантаження оператора ділення для комплексних чисел.
- **bool Complex::operator==(const Complex other) const**: Перевантаження оператора порівняння на рівність.

- **bool Complex::operator!=(const Complex other) const**: Перевантаження оператора порівняння на нерівність.
- **double Complex::modulus() const**: Метод для обчислення модуля комплексного числа.
- **std::ostream operator«(std::ostream os, const Complex c)**: Перевантаження оператора виведення на екран.
- **std::istream operator»(std::istream is, Complex c)**: Перевантаження оператора введення з клавіатури.
- **void Complex::saveToFile(std::ofstream file) const**: Метод для збереження комплексного числа в файл.
- **Complex Complex::readFromFile(std::ifstream file)**: Метод для зчитування комплексного числа з файлу.

4 Функції вводу/виводу для власноруч зроблених бібліотек

4.1 Функції вводу/виводу для C

Для реалізації вводу та виводу комплексних чисел на C використовуються наступні функції:

- **Complex readComplex()**: Ця функція зчитує комплексне число з клавіатури. Вона запитує у користувача введення дійсної та уявної частин числа і повертає їх у вигляді структури `Complex`.

```
Complex readComplex() {
    Complex a;
    printf("Enter real part: ");
    scanf("%lf", &a.real);
    printf("Enter imaginary part: ");
    scanf("%lf", &a.imag);
    return a;
}
```

- **void printComplex(Complex a)**: Ця функція виводить комплексне число у вигляді `a.real + a.imag i` на екран.

```
void printComplex(Complex a) {
    printf("Complex number: %.2f + %.2fi\n", a.real, a.imag);
}
```

- **void saveToFile(FILE *file, Complex a)**: Ця функція зберігає комплексне число у файл. Вона записує значення дійсної та уявної частин числа в текстовий файл.

```
void saveToFile(FILE *file, Complex a) {
    fprintf(file, "%.2f %.2f\n", a.real, a.imag);
}
```

- **Complex readFromFile(FILE *file)**: Ця функція зчитує комплексне число з файлу. Вона отримує значення дійсної та уявної частин числа з текстового файлу і повертає їх у вигляді структури `Complex`.

```
Complex readFromFile(FILE *file) {
    Complex a;
    fscanf(file, "%lf %lf", &a.real, &a.imag);
    return a;
}
```

4.2 Функції вводу/виводу для C++

У C++ для вводу та виводу комплексних чисел використовуються перевантажені оператори вводу/виводу.

- **std::istream operator»(std::istream is, Complex c)**: Перевантажений оператор для введення комплексного числа з клавіатури. Користувач вводить спочатку дійсну, а потім уявну частину числа.

```
std::istream& operator>>(std::istream& is, Complex& c) {
    std::cout << "Enter real part: ";
    is >> c.real;
    std::cout << "Enter imaginary part: ";
    is >> c.imag;
    return is;
}
```

- **std::ostream operator«(std::ostream os, const Complex c)**: Перевантажений оператор для виведення комплексного числа на екран. Комплексне число виводиться у форматі `a.real + a.imag i`.

```
std::ostream& operator<<(std::ostream& os, const Complex& c) {
    os << c.real << " + " << c.imag << "i";
    return os;
}
```

- **void Complex::saveToFile(std::ofstream file) const**:
- **void Complex::saveToFile(std::ofstream file) const**: Цей метод зберігає комплексне число в файл. Він записує дійсну та уявну частини числа у файл, використовуючи об'єкт `ofstream` для роботи з файлом.

```
void Complex::saveToFile(std::ofstream& file) const {
    file << real << " " << imag << std::endl;
}
```

- **Complex Complex::readFromFile(std::ifstream file)**: Цей метод зчитує комплексне число з файлу. Він зчитує два значення (дійсну та уявну частини) з файлу та повертає новий об'єкт типу `Complex`.

```
Complex Complex::readFromFile(std::ifstream& file) {
    double real, imag;
    file >> real >> imag;
    return Complex(real, imag);
}
```

5 Опис функцій у бібліотеках C та C++

5.1 Бібліотека C

У цій бібліотеці були реалізовані основні арифметичні операції з комплексними числами, а також функції для вводу/виводу даних.

- **Complex add(Complex a, Complex b)**: Функція для додавання двох комплексних чисел. Вона приймає два аргументи типу `Complex` і повертає їх суму.

```
Complex add(Complex a, Complex b) {
    Complex result = {a.real + b.real, a.imag + b.imag};
    return result;
}
```

- **Complex subtract(Complex a, Complex b)**: Функція для віднімання двох комплексних чисел. Вона приймає два аргументи типу `Complex` і повертає їх різницю.

```
Complex subtract(Complex a, Complex b) {
    Complex result = {a.real - b.real, a.imag - b.imag};
    return result;
}
```

- **Complex multiply(Complex a, Complex b)**: Функція для множення двох комплексних чисел. Вона реалізує формулу множення комплексних чисел, де дійсна частина та уявна частина обчислюються окремо.

```
Complex multiply(Complex a, Complex b) {
    Complex result = {a.real * b.real - a.imag * b.imag, a.rea
    return result;
}
```

- **Complex divide(Complex a, Complex b)**: Функція для ділення двох комплексних чисел. Вона реалізує формулу ділення комплексних чисел з обчисленням знаменника, що містить квадрат модуля другого числа.

```
Complex divide(Complex a, Complex b) {
    double denominator = b.real * b.real + b.imag * b.imag;
    Complex result = {
        (a.real * b.real + a.imag * b.imag) / denominator,
        (a.imag * b.real - a.real * b.imag) / denominator
    };
    return result;
}
```

- **double modulus(Complex a)**: Функція для обчислення модуля комплексного числа. Вона повертає квадратний корінь із суми квадратів дійсної та уявної частин.

```
double modulus(Complex a) {
    return sqrt(a.real * a.real + a.imag * a.imag);
}
```

- **void printComplex(Complex a)**: Функція для виведення комплексного числа на екран. Вона виводить дійсну та уявну частини комплексного числа у форматі "a + bi".

```
void printComplex(Complex a) {
    printf("Complex number: %.2f + %.2fi\n", a.real, a.imag);
}
```

- **Complex readComplex()**: Функція для введення комплексного числа з клавіатури. Вона зчитує дійсну та уявну частини комплексного числа від користувача.

```
Complex readComplex() {
    Complex a;
    printf("Enter real part: ");
    scanf("%lf", &a.real);
    printf("Enter imaginary part: ");
    scanf("%lf", &a.imag);
    return a;
}
```

- **void saveToFile(FILE *file, Complex a)**: Функція для збереження комплексного числа у файл. Вона записує дійсну та уявну частини числа у файл у форматі "a b".

```
void saveToFile(FILE *file, Complex a) {
    fprintf(file, "%.2f %.2f\n", a.real, a.imag);
}
```

- **Complex readFromFile(FILE *file)**: Функція для зчитування комплексного числа з файлу. Вона зчитує два числа (дійсну та уявну частину) та створює комплексне число.

```
Complex readFromFile(FILE *file) {
    Complex a;
    fscanf(file, "%lf %lf", &a.real, &a.imag);
    return a;
}
```

6 Опис імплементованих готових бібліотек на C та C++

6.1 Імплементовані бібліотеки на C

У C для роботи з комплексними числами існує стандартна бібліотека `<complex.h>`, яка надає базові функції для роботи з комплексними числа-

ми. Бібліотека підтримує основні арифметичні операції та інші операції, пов'язані з комплексними числами. Ось основні функції, що реалізовані в цій бібліотеці:

- **double complex addStd(double complex a, double complex b)**: Функція для додавання двох комплексних чисел. Вона приймає два аргументи типу `double complex` і повертає їх суму.

```
double complex addStd(double complex a, double complex b) {
    return a + b;
}
```

- **double complex subtractStd(double complex a, double complex b)**: Функція для віднімання двох комплексних чисел. Вона приймає два аргументи типу `double complex` і повертає їх різницю.

```
double complex subtractStd(double complex a, double complex b)
    return a - b;
}
```

- **double complex multiplyStd(double complex a, double complex b)**: Функція для множення двох комплексних чисел. Вона реалізує стандартну формулу множення комплексних чисел.

```
double complex multiplyStd(double complex a, double complex b)
    return a * b;
}
```

- **double complex divideStd(double complex a, double complex b)**: Функція для ділення двох комплексних чисел. Вона реалізує стандартну формулу ділення комплексних чисел.

```
double complex divideStd(double complex a, double complex b) {
    return a / b;
}
```

- **double modulus(double complex a)**: Функція для обчислення модуля комплексного числа. Вона використовує стандартну бібліотеку `cabs()` для обчислення модуля.

```
double modulus(double complex a) {
    return cabs(a);
}
```

- **void printComplex(double complex a)**: Функція для виведення комплексного числа. Вона виводить число у форматі "a + bi".

```
void printComplex(double complex a) {
    printf("Complex number: %.2f + %.2fi\n", creal(a), cimag(a
}
```

- **double complex readComplex()**: Функція для введення комплексного числа з клавіатури. Вона зчитує два числа для дійсної та уявної частини.

```
double complex readComplex() {
    double real, imag;
    printf("Enter real part: ");
    scanf("%lf", &real);
    printf("Enter imaginary part: ");
    scanf("%lf", &imag);
    return real + imag * I;
}
```

- **void saveToFile(FILE *file, double complex a)**: Функція для збереження комплексного числа у файл. Вона записує дійсну та уявну частини числа в файл.

```
void saveToFile(FILE *file, double complex a) {
    fprintf(file, "%.2f %.2f\n", creal(a), cimag(a));
}
```

- **double complex readFromFile(FILE *file)**: Функція для зчитування комплексного числа з файлу. Вона зчитує два числа для дійсної та уявної частини і створює комплексне число.

```
double complex readFromFile(FILE *file) {
    double real, imag;
    fscanf(file, "%lf %lf", &real, &imag);
    return real + imag * I;
}
```

6.2 Імплементовані бібліотеки на C++

У C++ для роботи з комплексними числами також існує стандартна бібліотека `<complex>`, яка надає багатий набір функцій та операторів для

роботи з комплексними числами. Ось основні функції, що реалізовані в цій бібліотеці:

- **std::complex<double> addStd(std::complex<double> a, std::complex<double> b)**: Функція для додавання двох комплексних чисел. Вона приймає два аргументи типу `std::complex<double>` і повертає їх суму.

```
std::complex<double> addStd(std::complex<double> a, std::compl
    return a + b;
}
```

- **std::complex<double> subtractStd(std::complex<double> a, std::complex<double> b)**: Функція для віднімання двох комплексних чисел. Вона приймає два аргументи типу `std::complex<double>` і повертає їх різницю.

```
std::complex<double> subtractStd(std::complex<double> a, std::
    return a - b;
}
```

- **std::complex<double> multiplyStd(std::complex<double> a, std::complex<double> b)**: Функція для множення двох комплексних чисел. Вона використовує стандартне множення комплексних чисел у C++.

```
std::complex<double> multiplyStd(std::complex<double> a, std::
    return a * b;
}
```

- **std::complex<double> divideStd(std::complex<double> a, std::complex<doubl b)**: Функція для ділення двох комплексних чисел. Вона реалізує стандартну формулу для ділення комплексних чисел.

```
std::complex<double> divideStd(std::complex<double> a, std::co
    return a / b;
}
```

- **double modulusStd(std::complex<double> a)**: Функція для обчислення модуля комплексного числа. Вона використовує стандартну функцію `std::abs()` для обчислення модуля комплексного числа.

```
double modulusStd(std::complex<double> a) {
    return std::abs(a);
}
```

- **void printComplexStd(std::complex<double> a)**: Функція для виведення комплексного числа на екран у форматі "a + bi".

```
void printComplexStd(std::complex<double> a) {
    std::cout << "Complex number: " << a.real() << " + " << a.
}
```

- **std::complex<double> readComplexStd()**: Функція для введення комплексного числа з клавіатури.

```
std::complex<double> readComplexStd() {
    double real, imag;
    std::cout << "Enter real part: ";
    std::cin >> real;
    std::cout << "Enter imaginary part: ";
    std::cin >> imag;
    return std::complex<double>(real, imag);
}
```

- **void saveToFileStd(std::ofstream file, std::complex<double> a)**: Функція для збереження комплексного числа у файл.

```
void saveToFileStd(std::ofstream& file, std::complex<double> a
    file << a.real() << " " << a.imag() << std::endl;
}
```

- **std::complex<double> readFromFileStd(std::ifstream file)**: Функція для зчитування комплексного числа з файлу.

```
std::complex<double> readFromFileStd(std::ifstream& file) {
    double real, imag;
    file >> real >> imag;
    return std::complex<double>(real, imag);
}
```

6.3 Бібліотека C++

У цій бібліотеці були реалізовані подібні функції, але вони обгорнуті в клас `Complex`, що дає змогу використовувати переваги об'єктно-орієнтованого програмування, зокрема перевантаження операторів та інкапсуляцію.

- **Complex::Complex()**: Конструктор за замовчуванням для створення комплексного числа з нульовими значеннями дійсної та уявної частини.

```
Complex::Complex() : real(0), imag(0) {}
```

- **Complex::Complex(double real, double imag)**: Конструктор з параметрами для створення комплексного числа з заданими дійсною та уявною частинами.

```
Complex::Complex(double real, double imag) : real(real), imag(
```

- **Complex Complex::operator+(const Complex other) const**: Оператор додавання для комплексних чисел. Додає дійсні та уявні частини двох комплексних чисел.

```
Complex Complex::operator+(const Complex& other) const {
    return Complex(real + other.real, imag + other.imag);
}
```

- **Complex Complex::operator-(const Complex other) const**: Оператор віднімання для комплексних чисел. Віднімає дійсні та уявні частини двох комплексних чисел.

```
Complex Complex::operator-(const Complex& other) const {
    return Complex(real - other.real, imag - other.imag);
}
```

- **Complex Complex::operator*(const Complex other) const**: Оператор множення для комплексних чисел. Реалізує стандартне множення комплексних чисел.

```
Complex Complex::operator*(const Complex& other) const {
    return Complex(real * other.real - imag * other.imag, real
}
```

- **Complex Complex::operator/(const Complex other) const**: Оператор ділення для комплексних чисел. Використовує формулу для ділення комплексних чисел з обчисленням знаменника.

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

- **bool Complex::operator==(const Complex other) const**: Оператор рівності для порівняння комплексних чисел. Повертає `true`, якщо обидва числа рівні.

```
bool Complex::operator==(const Complex& other) const {
    return real == other.real && imag == other.imag;
}
```

- **bool Complex::operator!=(const Complex other) const**: Оператор нерівності для порівняння комплексних чисел. Повертає `true`, якщо обидва числа різні.

```
bool Complex::operator!=(const Complex& other) const {
    return !(*this == other);
}
```

- **double Complex::modulus() const**: Метод для обчислення модуля комплексного числа. Використовує стандартну формулу для обчислення модуля.

```
double Complex::modulus() const {
    return sqrt(real * real + imag * imag);
}
```

- **std::ostream operator«(std::ostream os, const Complex c)**: Перевантажений оператор виведення для виведення комплексного числа у форматі "a + bi".

```
std::ostream& operator<<(std::ostream& os, const Complex& c) {
    os << c.real << " + " << c.imag << "i";
    return os;
}
```

- **std::istream operator»(std::istream is, Complex c)**: Перевантажений оператор введення для введення комплексного числа з клавіатури.

```
std::istream& operator>>(std::istream& is, Complex& c) {
    std::cout << "Enter real part: ";
    is >> c.real;
    std::cout << "Enter imaginary part: ";
    is >> c.imag;
    return is;
}
```

- **void Complex::saveToFile(std::ofstream file) const**: Функція для збереження комплексного числа у файл.

```
void Complex::saveToFile(std::ofstream& file) const {
    file << real << " " << imag << std::endl;
}
```

- **Complex Complex::readFromFile(std::ifstream file)**: Функція для зчитування комплексного числа з файлу.

```
Complex Complex::readFromFile(std::ifstream& file) {
    double real, imag;
    file >> real >> imag;
    return Complex(real, imag);
}
```

7 Опис реалізації

7.1 Огляд реалізації коду

Цей код представляє клас `Matrix`, який вирішує системи лінійних рівнянь за допомогою алгоритму *Швидкого Перетворення Фур'є (FFT)*. Він здійснює збереження, маніпулювання та вирішення рівнянь, представлених у вигляді матриць, за допомогою комплексних чисел (які зберігаються як `ComplexType`). Клас надає функціональність для розв'язання системи як безпосередньо з векторів коефіцієнтів та констант, так і шляхом зчитування цих значень з файлу.

7.2 Основні компоненти коду

7.2.1 Конструктор класу `Matrix`

Є два конструктора в класі `Matrix`:

- **З векторів**: Цей конструктор приймає два вектори — `coefficients` (коефіцієнти) та `constants` (константи), які представляють матрицю коефіцієнтів і констант системи лінійних рівнянь відповідно. Він виконує перевірки, щоб переконатися, що жоден з векторів не порожній та що їхні розміри однакові.
- **З файлу**: Другий конструктор ініціалізує матрицю з файлу. Він зчитує коефіцієнти та константи, очікуючи від файлу наступний формат:

coefficients: value1 value2 ...
constants: value1 value2 ...

Виконуються необхідні перевірки, такі як перевірка формату файлу та відповідність розмірів векторів.

7.2.2 Функція `rowToColumn`

Ця функція приймає рядковий вектор (`firstRow`) і перетворює його на стовпцевий вектор, перевертаючи всі елементи, крім першого. Ця операція корисна для маніпулювання матрицями в алгоритмах, що вимагають представлення у вигляді стовпців.

7.2.3 Швидке Перетворення Фур'є

- `FastFourierTransform`: Функція `fastFourierTransform` виконує FFT для вектора. Це рекурсивна функція, яка розділяє вхідний вектор на парні та непарні елементи, застосовує до них FFT і потім об'єднує їх назад. Розмір вектора повинен бути степенем двійки, що перевіряється в реалізації. FFT є важливим інструментом для ефективного вирішення систем лінійних рівнянь через перехід у частотну область.

- `inverseF`: Функція `inverseFFT` інвертує операцію FFT. Спочатку вона бере комплексно спряжені елементи вектора, потім застосовує до них FFT, і в кінці нормалізує значення, ділячи на розмір вектора. Це дозволяє відновити рішення у часовій області.

7.2.4 Розв'язок системи (`solve`)

Метод `solve` вирішує систему лінійних рівнянь за допомогою FFT. Основний процес виглядає так:

1. Спочатку він перетворює рядковий вектор коефіцієнтів на стовпцевий за допомогою методу `rowToColumn`.

2. Потім застосовує FFT до векторів коефіцієнтів і констант.

3. Обчислює рішення у частотній області, ділячи FFT констант на FFT коефіцієнтів.

4. Наприкінці застосовує обернене FFT для отримання рішення у часовій області.

Цей метод повертає рішення у вигляді вектора комплексних чисел.

7.2.5 Виведення рішення (`solveAndOutput`)

Метод `solveAndOutput` не тільки вирішує систему, але й виводить рішення та час виконання в файл. Спочатку він вимірює час, витрачений на розв'язання системи, використовуючи бібліотеку `std::chrono`. Після розв'язання системи перевіряється, чи задовольняє отримане рішення перший рядок рівнянь.

7.3 Висновок по розділу 7

Клас `Matrix` забезпечує ефективне розв'язання систем лінійних рівнянь великого розміру, використовуючи властивості циркулюючих матриць. Застосування ШПФ значно знижує обчислювальну складність від $O(n^3)$ до $O(n \log n)$, що є суттєвою перевагою для систем великого розміру. Реалізація формул Кардано та Ферарі в C++ з використанням комплексних чисел

8 Реалізація формул Кардано та Ферарі в C++ з використанням комплексних чисел

У чисельних методах для розв'язування рівнянь вищих степенів, таких як кубічні та квартичні рівняння, широко використовуються формули Кардано та Ферарі. Ці методи дозволяють розв'язувати рівняння без застосування чисельних методів. Однак через можливість отримання комплексних коренів, для їх обчислення в програмуванні необхідно використовувати бібліотеки для роботи з комплексними числами. У нашій реалізації ми використовуємо стандартну бібліотеку C++ `std::complex`, що дозволяє зручно працювати з комплексними числами при обчисленні коренів рівнянь.

8.1 Формула Кардано для кубічного рівняння

Загальне кубічне рівняння має вигляд:

$$ax^3 + bx^2 + cx + d = 0$$

Щоб розв'язати це рівняння, спочатку виконується перетворення рівняння в вигляд $x^3+px+q = 0$, де p та q обчислюються через коефіцієнти початкового рівняння. Далі розв'язок отримується за допомогою формули Кардано, яка включає використання кубічних коренів та дискримінанту для знаходження коренів рівняння.

Основною ідеєю формули Кардано є застосування розв'язків через кубічні корені, що дозволяє знайти як дійсні, так і комплексні корені рівняння. У випадку наявності комплексних коренів, використовується стандартна бібліотека для роботи з комплексними числами `std::complex`, що дозволяє точно обчислити корені рівняння навіть у складних випадках.

8.2 Формула Ферарі для квартичного рівняння

Загальне квартичне рівняння має вигляд:

$$ax^4 + bx^3 + cx^2 + dx + e = 0$$

Формула Ферарі дозволяє звести квартичне рівняння до кубічного, яке потім можна розв'язати за допомогою формули Кардано. Це досягається шляхом введення нових змінних, що спрощує рівняння і дозволяє обчислити корені за допомогою класичних методів для кубічних рівнянь.

Процес зведення квартичного рівняння до кубічного є основним етапом в реалізації методу Ферарі. Після цього задача стає подібною до кубічного рівняння, і можна застосовувати ті ж самі техніки для розв'язання. Також, при роботі з комплексними коренями, використовуються засоби для роботи з комплексними числами, що дозволяє точно обчислити всі можливі корені.

8.3 Узагальнення по формулам

Реалізація формул Кардано та Ферарі на мові програмування C++ показує, як ефективно використовувати стандартну бібліотеку `std::complex` для обчислення коренів кубічних і квартичних рівнянь, включаючи випадки з комплексними коренями. Обидва методи, застосовані у програмуванні, дають можливість вирішити рівняння за допомогою алгебраїчних операцій, уникаючи чисельних методів, що робить їх корисними для теоретичних розрахунків і програмних реалізацій.

9 Висновок

У результаті проведеного дослідження було реалізовано дві бібліотеки для роботи з комплексними числами: на C та на C++. Кожна з цих реалізацій включала базові функції для арифметичних операцій над комплексними числами, а також функції для вводу та виводу чисел у файл і з файлу.

Для порівняння швидкості роботи цих бібліотек було створено тестовий файл `cross_tests.cpp`, який дозволяє виміряти час виконання операцій для кожної реалізації. Результати показали, що для простих операцій (додавання, віднімання) на C++ швидкість виконання дещо вища через використання вбудованих операцій та класів, які автоматизують деякі процеси. Однак для складніших обчислень, що включають великі об'єми даних, різниця у швидкості між C і C++ стає менш значущою, оскільки обидва методи оптимізовані для таких завдань.

Загалом, стандартна бібліотека для роботи з комплексними числами в C++ пропонує більш високий рівень абстракції, зручність та простоту використання. Однак, для більш специфічних задач або при роботі з обмеженими ресурсами, реалізація на C може бути більш ефективною.