

**LAPORAN TUGAS MODUL 2 PROGRAMMING
PROGRAM BEBAS DALAM BAHASA C++ YANG MENERAPKAN
KONSEP OOP (OBJECT-ORIENTED PROGRAMMING), INVERSE
KINEMATICS, DAN FILE HANDLING**

Github : <https://github.com/kveeyv/modul-2-robocon>

**Laporan ini disusun untuk memenuhi tugas modul 2 *internship* Robocon
2024**



ITS ROBOCON

Disusun Oleh :

Nama : M. MA'RUF QOMARUDDIN KAFI

NRP : 5002241095

Divisi : *Programming*

**ROBOCON ITS TEAM
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2024**

BAB I

DESKRIPSI

A. Deskripsi Umum

Laporan ini membahas empat program yang dikembangkan dalam bahasa C++, masing-masing dengan fokus yang berbeda namun saling terkait dalam penerapan konsep pemrograman berorientasi objek (OOP) dan inverse kinematics.

1. Program 1: KinematicsOmniBot

Program ini dirancang untuk menghitung kecepatan motor pada robot omni berdasarkan kecepatan translasi X, Y, dan rotasi Z yang diberikan oleh pengguna. Menggunakan konsep OOP, program ini menyimpan data ke dalam kelas Motor dan OmniRobot, yang menangani perhitungan dan tampilan hasil kecepatan motor. Pengguna dapat melihat rumus yang digunakan dalam perhitungan, serta hasil akhir kecepatan setiap motor.

2. Program 2: Question Answering Bot (ChatRoboCon)

Program ini berfungsi sebagai bot yang dapat menjawab pertanyaan dari pengguna. Menggunakan pemrograman berorientasi objek, program ini memuat pertanyaan dan jawaban dari file dan menyimpannya kembali setelah pertanyaan baru ditambahkan. Program ini menerapkan pencocokan kata kunci untuk memberikan jawaban yang relevan terhadap pertanyaan yang diajukan.

3. Program 3: Kinematics Data Logger (KinematicsAnalytics)

Program ini menggabungkan perhitungan kecepatan motor dan kemampuan untuk menyimpan hasil perhitungan ke dalam file. Pengguna dapat memasukkan kecepatan translasi, dan program akan menghitung kecepatan motor serta menyimpan input dan hasil ke dalam file teks. Pengguna juga dapat memilih untuk membaca hasil yang disimpan sebelumnya, meningkatkan kemampuan dokumentasi.

4. Program 4: Enhanced Kinematics (modul2robocon)

Program ini memperluas fungsionalitas program sebelumnya dengan penanganan format penyimpanan yang lebih variatif, termasuk opsi untuk menyimpan data dalam format JSON atau teks biasa. Program ini juga mencakup antarmuka

pengguna yang interaktif, yang memungkinkan pengguna untuk melakukan beberapa perhitungan kinematics dalam satu sesi dan mengelola hasilnya dengan lebih baik.

Keempat program ini bertujuan untuk memperdalam pemahaman mahasiswa mengenai pemrograman berorientasi objek, pengolahan data, dan penerapan konsep inverse kinematics dalam pengembangan aplikasi robotika sederhana yang dimana program tersebut saya simpan ke dalam repository github : <https://github.com/kveey/modul-2-robocon>

Adapun sebagai sumber referensi saya mendapatkan beberapa pengertian mengenai materi ini, beberapa diantaranya adalah :

1. Definisi Pemrograman Berorientasi Objek (OOP)

Menurut Bjarne Stroustrup dalam bukunya *"The C++ Programming Language" (4th Edition)*, pemrograman berorientasi objek adalah paradigma pemrograman yang menggunakan konsep objek dan kelas untuk membagi program menjadi bagian-bagian yang lebih kecil dan modular, yang memudahkan pengelolaan dan pemeliharaan kode.

(Sumber: Stroustrup, Bjarne. *The C++ Programming Language*. Addison-Wesley, 2013.)

2. Definisi Kelas

Menurut Herbert Schildt dalam bukunya *"C++: The Complete Reference (4th Edition)"*, kelas adalah cetak biru untuk membuat objek, yang mendefinisikan atribut dan metode yang dapat digunakan oleh objek tersebut. Kelas memungkinkan pengorganisasian kode yang lebih baik dan penggunaan kembali kode dalam program.

(Sumber: Schildt, Herbert. *C++: The Complete Reference*. McGraw-Hill, 2014.)

3. Definisi Objek

Menurut Stanley B. Lippman dalam bukunya *"C++ Primer (5th Edition)"*, objek adalah instance dari kelas yang memiliki data dan metode yang dapat diakses untuk melakukan tugas tertentu. Objek adalah komponen utama dalam pemrograman berorientasi objek.

(Sumber: Lippman, Stanley B. *C++ Primer*. Addison-Wesley, 2012.)

4. Definisi Inverse Kinematics

Menurut "*Introduction to Robotics: Mechanics and Control*" oleh John J. Leonard dan P. R. Kumar, inverse kinematics adalah proses menentukan gerakan atau sudut yang diperlukan untuk mencapai posisi tertentu dari bagian robot. Ini penting dalam robotika untuk menentukan bagaimana robot harus bergerak untuk mencapai tujuan tertentu.

(Sumber: Leonard, John J., dan P. R. Kumar. *Introduction to Robotics: Mechanics and Control*. Pearson, 2012.)

5. Definisi Input dan Output

Menurut Paul Deitel dan Harvey Deitel dalam bukunya "*C++ How to Program (10th Edition)*", input-output adalah mekanisme yang digunakan oleh program untuk menerima data dari pengguna (input) dan menampilkan hasilnya (output). C++ menyediakan berbagai metode seperti `cin` dan `cout` untuk mengelola input dan output.

(Sumber: Deitel, Paul, dan Harvey Deitel. *C++ How to Program*. Pearson Education, 2016.)

6. Definisi Fungsi (Method)

Menurut Bjarne Stroustrup dalam bukunya "*The C++ Programming Language*" (4th Edition), fungsi (atau metode) adalah blok kode yang dirancang untuk melakukan tugas tertentu dan dapat dipanggil dari bagian lain dalam program. Fungsi meningkatkan modularitas dan penggunaan kembali kode dalam pemrograman.

(Sumber: Stroustrup, Bjarne. *The C++ Programming Language*. Addison-Wesley, 2013.)

7. Definisi Struktur Kontrol (If-Else, Looping)

Menurut Herbert Schildt dalam bukunya "*C++: The Complete Reference (4th Edition)*", struktur kontrol adalah instruksi yang digunakan untuk mengontrol alur eksekusi program berdasarkan kondisi tertentu. C++ mendukung berbagai struktur kontrol, termasuk if-else untuk pengambilan keputusan dan perulangan (looping) untuk mengeksekusi blok kode berulang kali.

(Sumber: Schildt, Herbert. *C++: The Complete Reference*. McGraw-Hill, 2014.)

8. Definisi Penyimpanan Data (File Handling)

Menurut Paul Deitel dan Harvey Deitel dalam bukunya "*C++ How to Program (10th Edition)*", file handling dalam C++ mencakup metode dan kelas untuk membaca dan menulis data ke file. C++ menyediakan kelas seperti `fstream` untuk menangani operasi file secara efisien.

(Sumber: Deitel, Paul, dan Harvey Deitel. *C++ How to Program*. Pearson Education, 2016.)

9. Definisi Multi-threading

Menurut Anthony Williams dalam bukunya "*C++ Concurrency in Action*" (2nd Edition), multi-threading adalah kemampuan untuk menjalankan beberapa thread (jalur eksekusi) secara bersamaan dalam program. Ini memungkinkan program untuk melakukan beberapa tugas secara simultan, meningkatkan efisiensi dan responsivitas.

(Sumber: Williams, Anthony. *C++ Concurrency in Action*. Manning Publications, 2019.)

10. Definisi Struktur Data

Menurut Mark Allen Weiss dalam bukunya "*Data Structures and Algorithm Analysis in C++ (4th Edition)*", struktur data adalah cara organisasi dan penyimpanan data dalam program untuk memungkinkan akses dan modifikasi data yang efisien. C++ menyediakan berbagai struktur data, termasuk array, linked lists, dan trees.

(Sumber: Weiss, Mark Allen. *Data Structures and Algorithm Analysis in C++*. Addison-Wesley, 2013.)

11. Dll.

B. Deskripsi Lanjutan

1. Program 1: KinematicsOmniBot

- **Deklarasi Library dan Definisi Konstanta**

Program ini menggunakan library `iostream` untuk input-output dan `cmath` untuk perhitungan matematika. Terdapat juga konstanta `PI` dan `RAD` untuk perhitungan sudut.

- **Kelas Motor**

Kelas `Motor` digunakan untuk membuat objek motor. Kelas ini memiliki dua atribut:

- `theta`: Sudut roda dalam derajat.
- `speed`: Kecepatan motor.

Kelas ini memiliki dua metode:

- `calculateSpeed`: Menghitung kecepatan motor berdasarkan input yang diberikan.
- `displaySpeed`: Menampilkan informasi kecepatan motor.

- **Kelas OmniRobot**

Kelas `OmniRobot` berisi empat motor dengan sudut yang berbeda. Kelas ini mengatur perhitungan dan tampilan kecepatan motor:

- `calculateMotorSpeed`: Menghitung kecepatan untuk semua motor.
- `displayMotorSpeeds`: Menampilkan hasil kecepatan motor.
- `displayFormula`: Menampilkan rumus yang digunakan untuk perhitungan.

- **Kelas UserInterface**

Kelas `UserInterface` mengelola interaksi dengan pengguna. Kelas ini memiliki dua metode:

- `welcomeMessage`: Menampilkan pesan sambutan.
- `getInput`: Mengambil input kecepatan dari pengguna.

- **Fungsi main()**

Fungsi `main` adalah tempat program dimulai. Dalam fungsi ini:

- Objek `UserInterface` dan `OmniRobot` dibuat.
- Menampilkan pesan sambutan kepada pengguna.
- Menampilkan rumus yang digunakan.
- Mengambil input kecepatan dari pengguna.
- Menghitung kecepatan motor berdasarkan input.
- Menampilkan hasil kecepatan motor.

- **Penutupan Program**

- Program selesai setelah menampilkan semua hasil.

2. Program 2: Question Answering Bot (ChatRoboCon)

- **Deklarasi Library dan Preprocessor Directives**

Program ini menggunakan beberapa library seperti `iostream` untuk input-output, `fstream` untuk pengelolaan file, `unordered_map` untuk menyimpan pertanyaan dan jawaban, serta `string`, `algorithm`, dan `vector` untuk manipulasi string dan data. Terdapat juga pengaturan untuk mendukung fungsi `sleep` pada sistem operasi Windows dan Unix.

- **Kelas QuestionLoader**

Kelas `QuestionLoader` bertanggung jawab untuk memuat dan menyimpan pertanyaan serta jawaban dari dan ke file. Kelas ini memiliki dua metode utama:

- `loadQuestionsFromFile`: Memuat pertanyaan dan jawaban dari file ke dalam map. Jika file tidak ada, akan ditampilkan pesan kesalahan.
- `saveQuestionToFile`: Menyimpan pertanyaan dan jawaban baru ke dalam file.

- **Kelas KeywordMatcher**

Kelas `KeywordMatcher` digunakan untuk memeriksa apakah input pengguna mengandung kata kunci tertentu. Kelas ini memiliki satu metode:

- `containsKeyword`: Memeriksa apakah input mengandung salah satu kata kunci dari daftar yang diberikan.

- **Kelas QuestionAnsweringBot**

Kelas ini adalah inti dari program yang menangani interaksi dengan pengguna. Kelas ini memiliki atribut:

- `name`: Nama bot.
- `learned_questions_and_answers`: Map untuk menyimpan pertanyaan dan jawaban yang telah dipelajari.
- `filename`: Nama file tempat pertanyaan dan jawaban disimpan.

Kelas ini memiliki beberapa metode:

- Konstruktor untuk menginisialisasi bot dan memuat pertanyaan dari file.

- `getAnswer`: Mengambil jawaban untuk pertanyaan yang diajukan pengguna. Jika pertanyaan tidak dikenali, bot akan memberikan jawaban standar dan meminta pengguna untuk mengajarkan jawaban baru.
- `learnNewQuestion`: Menyimpan pertanyaan baru dan jawabannya ke dalam map dan file.

- **Fungsi `main()`**

Fungsi `main` adalah tempat program dimulai. Dalam fungsi ini:

- Objek `QuestionAnsweringBot` dibuat dengan nama bot dan nama file.
- Program memasuki loop untuk menerima input pertanyaan dari pengguna.
- Jika pengguna mengetik 'exit', program akan berhenti.
- Bot memberikan jawaban berdasarkan input, dan jika tidak tahu, meminta pengguna untuk mengajarkan jawaban baru.

- **Penutupan Program**

Program selesai setelah pengguna memilih untuk keluar, menampilkan pesan perpisahan.

3. Program 3: Kinematics Data Logger (KinematicsAnalytics)

- **Deklarasi Library dan Definisi Konstanta**

Program ini menggunakan library `iostream` untuk input-output, `cmath` untuk perhitungan matematika, dan `fstream` untuk pengelolaan file.

Konstanta `PI` dan `RAD` dideklarasikan untuk digunakan dalam perhitungan sudut.

- **Fungsi `saveDataToFile`**

Fungsi ini digunakan untuk menyimpan data ke dalam file `motor_output.txt`. Fungsi ini mencakup:

- Menyimpan header dan informasi input kecepatan.
- Menyimpan hasil kecepatan untuk setiap motor berdasarkan rumus kinematics.

- **Fungsi `readDataFromFile`**

Fungsi ini membaca data dari file `motor_output.txt` dan menampilkan

isi file ke konsol. Jika file tidak dapat dibuka, program akan menampilkan pesan kesalahan.

- **Fungsi `calculateMotorSpeed`**

Fungsi ini menghitung kecepatan motor berdasarkan input kecepatan translasi X, Y, dan rotasi Z yang diberikan oleh pengguna. Menggunakan rumus kinematics, fungsi ini menghitung kecepatan untuk empat motor dengan sudut yang berbeda.

- **Fungsi `displayMotorSpeeds`**

Fungsi ini menampilkan hasil kecepatan motor ke konsol, termasuk sudut setiap motor dan rumus yang digunakan.

- **Fungsi `displayFormula`**

Fungsi ini menampilkan rumus kinematics yang digunakan untuk perhitungan. Rumus ini menjelaskan bagaimana kecepatan motor dihitung berdasarkan sudut roda dan kecepatan translasi.

- **Fungsi `main()`**

Fungsi `main` adalah tempat program dimulai. Dalam fungsi ini:

- Pesan sambutan ditampilkan kepada pengguna.
- Rumus yang digunakan ditampilkan.
- Pengguna diminta untuk memasukkan kecepatan translasi X, Y, dan rotasi Z.
- Kecepatan motor dihitung dan disimpan ke dalam file.
- Hasil kecepatan motor ditampilkan kepada pengguna.
- Pengguna ditanya apakah ingin membaca hasil yang disimpan di file.

- **Penutupan Program**

Program diakhiri setelah semua proses selesai dan menampilkan hasil yang diperlukan.

4. Program 4: Enhanced Kinematics (modul2robocon)

- **Deklarasi Library dan Definisi Konstanta**

Program ini menggunakan beberapa library, yaitu `iostream` untuk input-output, `cmath` untuk perhitungan matematika, dan `fstream` untuk pengelolaan file. Konstanta `PI` dan `RAD` dideklarasikan untuk digunakan dalam perhitungan sudut.

- **Kelas Motor**

Kelas `Motor` digunakan untuk mendefinisikan objek motor dengan dua atribut:

- `theta`: Sudut roda dalam derajat.
- `speed`: Kecepatan motor yang dihitung berdasarkan input.
- Kelas ini memiliki dua metode:
- `calculateSpeed`: Menghitung kecepatan motor berdasarkan input kecepatan translasi X, Y, dan rotasi Z.
- `displaySpeed`: Menampilkan informasi kecepatan motor beserta sudutnya.

- **Kelas Kinematics**

Kelas `Kinematics` berisi empat objek `Motor`. Kelas ini mengatur perhitungan kecepatan motor dan menyimpan data ke dalam file:

- `calculateMotorSpeed`: Menghitung kecepatan untuk semua motor berdasarkan input yang diberikan.
- `displayMotorSpeeds`: Menampilkan hasil kecepatan motor ke konsol.
- `saveDataToFile`: Menyimpan input dan hasil ke file dalam format teks atau JSON.

- **Kelas UserInterface**

Kelas `UserInterface` bertanggung jawab untuk interaksi dengan pengguna. Kelas ini memiliki beberapa metode:

- `welcomeMessage`: Menampilkan pesan sambutan kepada pengguna.
- `getInput`: Mengambil input kecepatan dari pengguna dan melakukan validasi input.

- **Fungsi main()**

Fungsi `main` adalah tempat program dimulai. Dalam fungsi ini:

- Objek `UserInterface` dan `Kinematics` dibuat.
- Pesan sambutan ditampilkan kepada pengguna.
- Rumus yang digunakan untuk perhitungan ditampilkan.
- Pengguna diminta untuk memasukkan kecepatan translasi X, Y, dan rotasi Z.
- Kecepatan motor dihitung dan disimpan ke dalam file.
- Hasil kecepatan motor ditampilkan kepada pengguna.
- Pengguna ditanya apakah ingin membaca hasil yang disimpan di file.

- **Penutupan Program**

Program diakhiri setelah semua proses selesai dan menampilkan hasil yang diperlukan, dengan opsi untuk menghitung lagi atau membaca data dari file.

BAB II

SOURCE CODE

2.1 Program 1: KinematicsOmniBot

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

#define PI 3.14159265359
#define RAD PI / 180.0

using namespace std;

// ANSI color codes
#define RESET "\033[0m"
#define RED "\033[31m"
#define GREEN "\033[32m"
#define YELLOW "\033[33m"
#define BLUE "\033[34m"
#define MAGENTA "\033[35m"
#define CYAN "\033[36m"
#define WHITE "\033[37m"

class Motor {
public:
    float theta; // Sudut roda
    int speed; // Kecepatan motor

    Motor(float theta) : theta(theta), speed(0) {} // Constructor

    void calculateSpeed(float wheel_matrix[3], int speed_vector[3]) {
        speed = 0;
        for (int j = 0; j < 3; j++) {
            speed += wheel_matrix[j] * speed_vector[j];
        }
    }

    void displaySpeed(int motorIndex) const {
```

```

        cout << "Motor " << motorIndex << " (Theta = " << theta << "
derajat):" << endl;
        cout << MAGENTA << "V_" << motorIndex << " = cos(" << theta << ")
* X + sin(" << theta << ") * Y + Z" << RESET << endl;
        cout << "Kecepatan motor " << motorIndex << " : " << speed <<
endl << endl;
    }
};

class OmniRobot {
private:
    Motor motors[4]; // Array dari objek Motor
    float wheel_matrix[4][3]; // Matriks yang merepresentasikan orientasi
roda

public:
    OmniRobot() : motors{Motor(45.0), Motor(135.0), Motor(225.0),
Motor(315.0)} {
        // Menginisialisasi matriks orientasi roda omni
        for (int i = 0; i < 4; i++) {
            wheel_matrix[i][0] = cos(motors[i].theta * RAD);
            wheel_matrix[i][1] = sin(motors[i].theta * RAD);
            wheel_matrix[i][2] = 1; // Faktor konstanta
        }
    }

    void calculateMotorSpeed(int speed_vector[3]) {
        for (int i = 0; i < 4; i++) {
            motors[i].calculateSpeed(wheel_matrix[i], speed_vector);
        }
    }

    void displayMotorSpeeds() const {
        cout << GREEN << "Hasil perhitungan kecepatan motor:" << RESET <<
endl;
        for (int i = 0; i < 4; i++) {
            motors[i].displaySpeed(i);
        }
    }

    void displayFormula() const {
        cout << BLUE << "Rumus Inverse Kinematics:" << RESET << endl;
        cout << "V_i = cos(theta_i) * x + sin(theta_i) * y + z" << endl;
        cout << "Di mana theta_i adalah sudut roda ke-i dalam derajat."
<< endl << endl;
    }
};

```

```

class UserInterface {
public:
    void welcomeMessage() const {
        cout << CYAN << "Selamat datang di KinematicsOmniBot!" << RESET
        << endl;
        cout << "Disini kamu bisa memasukkan kecepatan translasi X, Y,
        dan rotasi Z." << endl;
        cout << "Program ini akan memberikan hasil terkait kecepatan
        setiap motor robot." << endl << endl;
    }

    void getInput(int input_speed[3]) const {
        cout << "Masukkan kecepatan translasi X: ";
        cin >> input_speed[0];
        cout << "Masukkan kecepatan translasi Y: ";
        cin >> input_speed[1];
        cout << "Masukkan kecepatan rotasi Z: ";
        cin >> input_speed[2];
    }
};

int main() {
    UserInterface ui;
    OmniRobot robot;
    int input_speed[3];

    // Pesan sambutan
    ui.welcomeMessage();

    // Tampilkan rumus yang digunakan
    robot.displayFormula();

    // Input kecepatan translasi X, Y, dan rotasi Z
    ui.getInput(input_speed);

    // Hitung kecepatan motor berdasarkan input kecepatan
    robot.calculateMotorSpeed(input_speed);

    // Tampilkan hasil kecepatan motor dan sudut
    robot.displayMotorSpeeds();

    return 0;
}

```

2.2 Program 2: Question Answering Bot (ChatRoboCon)

```
#include <iostream>
#include <fstream>
#include <unordered_map>
#include <string>
#include <algorithm>
#include <vector>
#include <ctime>
#ifdef _WIN32
    #include <windows.h> // Library untuk fungsi Sleep di Windows
#else
    #include <unistd.h> // Library untuk fungsi sleep di sistem Unix
#endif

// Kelas untuk memuat pertanyaan dan jawaban dari file serta menyimpan
// pertanyaan baru ke file
class QuestionLoader {
public:
    // Method untuk memuat pertanyaan dan jawaban yang sudah dipelajari
    // dari file
    static void loadQuestionsFromFile(const std::string& filename,
std::unordered_map<std::string, std::string>&
learned_questions_and_answers) {
        std::ifstream file(filename);
        // Jika file tidak ada, tampilkan pesan bahwa file belum ada
        if (!file) {
            std::cout << "File belum ada atau tidak dapat dibuka. Mulai
dari awal." << std::endl;
            return;
        }
        std::string question, answer;
        // Membaca setiap baris pertanyaan dan jawaban dari file
        while (std::getline(file, question) && std::getline(file,
answer)) {
            if (!question.empty() && !answer.empty()) {
                learned_questions_and_answers[toLowerCase(question)] =
answer; // Simpan ke dalam map dengan key lowercase
            }
        }
        file.close(); // Tutup file setelah selesai membaca
```

```

    }

    // Method untuk menyimpan pertanyaan dan jawaban yang baru dipelajari
    ke file
    static void saveQuestionToFile(const std::string& filename, const
std::string& question, const std::string& answer) {
        std::ofstream file(filename, std::ios::app); // Buka file dan
tambahkan data di akhir
        if (file.is_open()) {
            file << question << std::endl; // Simpan pertanyaan
            file << answer << std::endl; // Simpan jawaban
            file.close(); // Tutup file setelah selesai menulis
        }
    }

    // Helper function untuk mengubah string menjadi huruf kecil
    (lowercase)
    static std::string toLowerCase(const std::string& str) {
        std::string result = str;
        std::transform(result.begin(), result.end(), result.begin(),
::tolower); // Ubah setiap karakter menjadi lowercase
        return result;
    }

    // Helper function untuk memeriksa apakah input mengandung kata kunci
    tertentu
    static bool containsKeyword(const std::string& input, const
std::string& question) {
        return input.find(question) != std::string::npos; // Cek apakah
kata kunci ditemukan dalam input
    }
};

// Kelas untuk mencocokkan beberapa kata kunci dalam satu input
class KeywordMatcher {
public:
    // Method untuk memeriksa apakah input mengandung salah satu dari
    sekumpulan kata kunci
    static bool containsKeyword(const std::string& text, const
std::vector<std::string>& keywords) {
        for (const auto& keyword : keywords) {
            if (text.find(keyword) != std::string::npos) { // Jika
ditemukan kata kunci, return true
                return true;
            }
        }
        return false; // Jika tidak ditemukan kata kunci, return false
    }
};

```

```

    }
};

// Kelas utama bot yang menangani pertanyaan pengguna dan jawaban
class QuestionAnsweringBot {
private:
    std::string name; // Nama bot
    std::unordered_map<std::string, std::string>
learned_questions_and_answers; // Map untuk menyimpan pertanyaan dan
jawaban yang dipelajari
    std::string filename; // Nama file tempat menyimpan pertanyaan dan
jawaban

public:
    // Konstruktor bot, diinisialisasi dengan nama bot dan file
penyimpanan
    QuestionAnsweringBot(const std::string& bot_name, const std::string&
file): name(bot_name), filename(file) {
        std::cout << "Halo, namaku adalah " << name << "!" << std::endl;
        std::cout << "Tanya pertanyaan. Ketik 'exit' untuk keluar." <<
std::endl;
        QuestionLoader::loadQuestionsFromFile(filename,
learned_questions_and_answers); // Muat pertanyaan yang dipelajari dari
file
    }

    // Method untuk menjawab pertanyaan pengguna
    std::string getAnswer(const std::string& question) {
        std::string q_lower = QuestionLoader::toLowerCase(question); //
Ubah input pengguna menjadi lowercase

        // Cek apakah pertanyaan sudah dipelajari sebelumnya
        for (const auto& entry : learned_questions_and_answers) {
            if (QuestionLoader::containsKeyword(q_lower, entry.first)) {
                return entry.second; // Jika ditemukan kecocokan,
berikan jawaban yang sesuai
            }
        }

        // Cek kecocokan berdasarkan keyword yang sudah ditentukan
        if (KeywordMatcher::containsKeyword(q_lower, {"nama mu", "kamu
siapa"})) {
            return "Nama saya " + name + ".";
        } else if (KeywordMatcher::containsKeyword(q_lower, {"apa kabar",
"kabar"})) {
            return "Saya baik, terima kasih! Bagaimana denganmu?";

```



```

        } else if (KeywordMatcher::containsKeyword(q_lower, {"oop",
"object oriented"})) {
            return "OOP adalah Object Oriented Programming, sebuah
paradigma pemrograman.";
        } else if (KeywordMatcher::containsKeyword(q_lower, {"presiden
indonesia", "siapa presiden"})) {
            return "Presiden Indonesia saat ini adalah Joko Widodo.";
        } else if (KeywordMatcher::containsKeyword(q_lower, {"ibukota
indonesia", "ibukota"})) {
            return "Ibukota Indonesia adalah Jakarta.";
        } else if (KeywordMatcher::containsKeyword(q_lower, {"berikan
saya lelucon", "lelucon"})) {
            return "Mengapa sepeda tidak bisa berdiri sendiri? Karena dia
hanya punya dua roda!";
        } else if (KeywordMatcher::containsKeyword(q_lower, {"siapa
penemu robot", "penemu robot"})) {
            return "Penemu robot adalah Al-Jazari.";
        } else if (KeywordMatcher::containsKeyword(q_lower, {"apa warna
langit", "warna langit"})) {
            return "Warna langit biasanya biru pada siang hari.";
        } else if (KeywordMatcher::containsKeyword(q_lower, {"its
adalah", "taukah kamu its"})) {
            // Jawaban bertahap dengan penundaan
            std::cout << "Bot: ITS adalah Institut Teknologi terbaik di
Indonesia..." << std::endl;
            #ifdef _WIN32
                Sleep(2000); // Untuk Windows
            #else
                sleep(2); // Untuk Unix
            #endif
            return "Setelah ITB :)";
        } else if (KeywordMatcher::containsKeyword(q_lower, {"its
robocon", "taukah kamu its robocon"})) {
            return "ITS Robocon Team adalah tim riset robotika tertua,
terkeren, terketch, terhebat, tergantung, tercanggih, terinovatif,
terkuat, terjago, ter2 di ITS.";
        } else {
            return "Maaf, saya tidak tahu jawabannya.";
        }
    }

    // Method untuk belajar pertanyaan baru dari pengguna
    void learnNewQuestion(const std::string& question, const std::string&
answer) {
        learned_questions_and_answers[QuestionLoader::toLowerCase(questio
n)] = answer; // Simpan ke map

```

```

        QuestionLoader::saveQuestionToFile(filename, question,
answer); // Simpan juga ke file
    }
};

// Program utama
int main() {
    QuestionAnsweringBot bot("ChatRoboCon", "learned_questions.txt"); //
Inisialisasi bot dengan nama dan file

    while (true) {
        std::string question;
        std::cout << "Kamu: ";
        std::getline(std::cin, question); // Baca input dari pengguna

        // Jika pengguna mengetik 'exit', keluar dari loop
        if (question == "exit") {
            std::cout << "Bot: Terima kasih! Sampai jumpa!" << std::endl;
            break;
        }

        // Cek apakah input kosong
        if (question.empty()) {
            std::cout << "Input tidak boleh kosong. Silakan coba lagi."
<< std::endl;
            continue;
        }

        // Dapatkan jawaban dari bot
        std::string answer = bot.getAnswer(question);
        std::cout << "Bot: " << answer << std::endl;

        // Jika bot tidak tahu jawabannya, minta pengguna untuk
mengajarkan jawaban baru
        if (answer == "Maaf, saya tidak tahu jawabannya.") {
            std::cout << "Ajari saya jawaban untuk pertanyaan ini: ";
            std::string new_answer;
            std::getline(std::cin, new_answer);

            // Cek apakah jawaban yang diberikan valid
            if (!new_answer.empty()) {
                bot.learnNewQuestion(question, new_answer);
                std::cout << "Terima kasih! Saya sudah belajar jawaban
baru." << std::endl;
            } else {
                std::cout << "Jawaban tidak boleh kosong." << std::endl;
            }
        }
    }
}

```

```

    }
}

return 0;
}

```

2.3 Program 3: Kinematics Data Logger (KinematicsAnalytics)

```

#include <iostream>
#include <cmath>
#include <fstream>
#include <string>

#define PI 3.14159265359
#define RAD PI / 180.0

using namespace std;

// ANSI color codes
#define RESET "\033[0m"
#define RED "\033[31m"
#define GREEN "\033[32m"
#define YELLOW "\033[33m"
#define BLUE "\033[34m"
#define MAGENTA "\033[35m"
#define CYAN "\033[36m"
#define WHITE "\033[37m"

// Fungsi untuk menyimpan data ke file
void saveDataToFile(int input_speed[3], int motor_speeds[4]) {
    ofstream output_file("motor_output.txt");

    // Menyimpan header
    output_file << "=====\n";
    output_file << "          Kinematics Output      \n";
    output_file << "=====\n\n";

    output_file << "Input Kecepatan:\n";
    output_file << "X: " << input_speed[0] << ", Y: " << input_speed[1]
    << ", Z: " << input_speed[2] << "\n";
    output_file << "\nHasil Kecepatan Motor:\n";
    output_file << "-----\n";

    for (int i = 0; i < 4; i++) {
        output_file << "Motor " << i << " (Theta = " << (45.0 + i * 90)
        << " derajat):\n";
    }
}

```

```

        output_file << "V_" << i << " = cos(" << (45.0 + i * 90) << ") *
X + sin(" << (45.0 + i * 90) << ") * Y + Z\n";
        output_file << "Kecepatan motor " << i << " : " <<
motor_speeds[i] << "\n";
        output_file << "-----\n";
    }

    output_file.close();
}

// Fungsi untuk membaca data dari file
void readDataFromFile() {
    ifstream input_file("motor_output.txt");
    string line;

    if (input_file.is_open()) {
        cout << "\n===== \n";
        cout << "          Kinematics Output          \n";
        cout << "===== \n\n";

        while (getline(input_file, line)) {
            cout << line << endl;
        }
        input_file.close();
    } else {
        cout << "Tidak bisa membuka file!" << endl;
    }
}

// Fungsi untuk menghitung kecepatan motor
void calculateMotorSpeed(int input_speed[3], int motor_speeds[4]) {
    motor_speeds[0] = cos(45.0 * RAD) * input_speed[0] + sin(45.0 * RAD)
* input_speed[1] + input_speed[2];
    motor_speeds[1] = cos(135.0 * RAD) * input_speed[0] + sin(135.0 *
RAD) * input_speed[1] + input_speed[2];
    motor_speeds[2] = cos(225.0 * RAD) * input_speed[0] + sin(225.0 *
RAD) * input_speed[1] + input_speed[2];
    motor_speeds[3] = cos(315.0 * RAD) * input_speed[0] + sin(315.0 *
RAD) * input_speed[1] + input_speed[2];
}

// Fungsi untuk menampilkan hasil kecepatan motor
void displayMotorSpeeds(int motor_speeds[4]) {
    cout << GREEN << "Hasil perhitungan kecepatan motor:" << RESET <<
endl;
    for (int i = 0; i < 4; i++) {

```

```

        cout << "Motor " << i << " (Theta = " << (45.0 + i * 90) << "
derajat):" << endl;
        cout << MAGENTA << "V_" << i << " = cos(" << (45.0 + i * 90) <<
") * X + sin(" << (45.0 + i * 90) << ") * Y + Z" << RESET << endl;
        cout << "Kecepatan motor " << i << " : " << motor_speeds[i] <<
endl << endl;
    }
}

// Fungsi untuk menampilkan rumus kinematics
void displayFormula() {
    cout << BLUE << "Rumus Inverse Kinematics:" << RESET << endl;
    cout << "V_i = cos(theta_i) * x + sin(theta_i) * y + z" << endl;
    cout << "Di mana theta_i adalah sudut roda ke-i dalam derajat." <<
endl << endl;
}

int main() {
    int input_speed[3];
    int motor_speeds[4];

    // Pesan sambutan
    cout << CYAN << "Selamat datang di KinematicsOmniBot!" << RESET <<
endl;
    cout << "Disini kamu bisa memasukkan kecepatan translasi X, Y, dan
rotasi Z." << endl;
    cout << "Program ini akan memberikan hasil terkait kecepatan setiap
motor robot." << endl << endl;

    // Tampilkan rumus yang digunakan
    displayFormula();

    // Input kecepatan translasi X, Y, dan rotasi Z
    cout << "Masukkan kecepatan translasi X: ";
    cin >> input_speed[0];
    cout << "Masukkan kecepatan translasi Y: ";
    cin >> input_speed[1];
    cout << "Masukkan kecepatan rotasi Z: ";
    cin >> input_speed[2];

    // Hitung kecepatan motor berdasarkan input kecepatan
    calculateMotorSpeed(input_speed, motor_speeds);

    // Simpan input dan hasil ke file
    saveDataToFile(input_speed, motor_speeds);

    // Tampilkan hasil kecepatan motor

```

```

displayMotorSpeeds(motor_speeds);

// Tanyakan apakah pengguna ingin membaca file
char choice;
cout << "Apakah Anda ingin melihat hasil yang disimpan di file?
(y/n): ";
cin >> choice;

if (choice == 'y' || choice == 'Y') {
    readDataFromFile();
}

return 0;
}

```

2.4 Program 4: Enhanced Kinematics (modul2robocon)

```

#include <iostream>
#include <cmath>
#include <fstream>
#include <limits>
#include <string>
#include <unordered_map>
#include <algorithm>

#define PI 3.14159265359
#define RAD PI / 180.0

using namespace std;

// ANSI color codes
#define RESET "\033[0m"
#define GREEN "\033[32m"
#define MAGENTA "\033[35m"
#define CYAN "\033[36m"
#define BLUE "\033[34m"

// Kelas Motor
class Motor {
public:
    float theta; // Sudut roda
    int speed;    // Kecepatan motor

    Motor(float theta) : theta(theta), speed(0) {} // Constructor

    void calculateSpeed(int input_speed[3]) {

```

```

        speed = cos(theta * RAD) * input_speed[0] + sin(theta * RAD) *
input_speed[1] + input_speed[2];
    }

    void displaySpeed(int motorIndex) const {
        cout << "Motor " << motorIndex << " (Theta = " << theta << "
derajat):" << endl;
        cout << MAGENTA << "V_" << motorIndex << " = cos(" << theta << ")
* X + sin(" << theta << ") * Y + Z" << RESET << endl;
        cout << "Kecepatan motor " << motorIndex << " : " << speed <<
endl << endl;
    }
};

// Kelas untuk menghitung kinematics
class Kinematics {
private:
    Motor motors[4]; // Array dari objek Motor
public:
    Kinematics() : motors{Motor(45.0), Motor(135.0), Motor(225.0),
Motor(315.0)} {}

    void calculateMotorSpeed(int input_speed[3]) {
        for (int i = 0; i < 4; i++) {
            motors[i].calculateSpeed(input_speed);
        }
    }

    void displayMotorSpeeds() const {
        cout << GREEN << "Hasil perhitungan kecepatan motor:" << RESET <<
endl;
        for (int i = 0; i < 4; i++) {
            motors[i].displaySpeed(i);
        }
    }

    bool saveDataToFile(int input_speed[3], const string& format) const {
        string filename = (format == "json") ? "motor_output.json" :
"motor_output.txt";
        ofstream output_file(filename);

        // Memeriksa apakah file berhasil dibuka
        if (!output_file.is_open()) {
            cout << "Error: Tidak bisa membuka file untuk menyimpan
data!" << endl;
            return false; // Keluar dari fungsi jika tidak dapat membuka
file

```

```

    }

    if (format == "plain") {
        // Menyimpan header
        output_file << "=====\n";
        output_file << "          Kinematics Output          \n";
        output_file << "=====\n\n";

        output_file << "Input Kecepatan:\n";
        output_file << "X: " << input_speed[0] << ", Y: " <<
input_speed[1] << ", Z: " << input_speed[2] << "\n";
        output_file << "\nHasil Kecepatan Motor:\n";
        output_file << "-----\n";

        for (int i = 0; i < 4; i++) {
            output_file << "Motor " << i << " (Theta = " << (45.0 + i
* 90) << " derajat):\n";
            output_file << "V_" << i << " = cos(" << (45.0 + i * 90)
<< ") * X + sin(" << (45.0 + i * 90) << ") * Y + Z\n";
            output_file << "Kecepatan motor " << i << " : " <<
motors[i].speed << "\n";
            output_file << "-----\n";
        }
    } else if (format == "json") {
        // Menyimpan dalam format JSON
        output_file << "{\n";
        output_file << "  \"Input Kecepatan\": {\n";
        output_file << "    \"X\": " << input_speed[0] << ",\n";
        output_file << "    \"Y\": " << input_speed[1] << ",\n";
        output_file << "    \"Z\": " << input_speed[2] << "\n";
        output_file << "  },\n";
        output_file << "  \"Hasil Kecepatan Motor\": [\n";

        for (int i = 0; i < 4; i++) {
            output_file << "    {\n";
            output_file << "      \"Motor\": " << i << ",\n";
            output_file << "      \"Theta\": " << (45.0 + i * 90) <<
",\n";
            output_file << "      \"Kecepatan\": " << motors[i].speed
<< "\n";
            output_file << "    }" << (i < 3 ? ", " : "") << "\n"; //
Tambah koma kecuali pada item terakhir
        }

        output_file << "  ]\n";
        output_file << "}\n";
    } else {

```



```

        cout << "Format tidak dikenali!" << endl;
    }

    output_file.close();
    return true; // Berhasil menyimpan data
}

void readDataFromFile(const string& filename) const {
    ifstream input_file(filename);
    string line;

    if (input_file.is_open()) {
        cout << "\n=====\\n";
        cout << "          Kinematics Output          \\n";
        cout << "=====\\n\\n";

        while (getline(input_file, line)) {
            // Menampilkan baris dengan warna
            if (line.find("Motor") != string::npos) {
                cout << GREEN << line << RESET << endl; // Warna
hijau untuk baris motor
            } else if (line.find("V_") != string::npos) {
                cout << MAGENTA << line << RESET << endl; // Warna
magenta untuk baris kecepatan
            } else {
                cout << line << endl; // Baris biasa
            }
        }
        input_file.close();
    } else {
        cout << "Tidak bisa membuka file!" << endl;
    }
}

};

// Kelas untuk antarmuka pengguna
class UserInterface {
public:
    void welcomeMessage() const {
        cout << CYAN << "Selamat datang di KinematicsOmniBot!" << RESET
<< endl;
        cout << "Disini kamu bisa memasukkan kecepatan translasi X, Y,
dan rotasi Z." << endl;
        cout << "Program ini akan memberikan hasil terkait kecepatan
setiap motor robot." << endl << endl;
    }
}

```

```

void getInput(int input_speed[3]) const {
    for (int i = 0; i < 3; i++) {
        while (true) {
            cout << "Masukkan kecepatan translasi " << (i == 0 ? "X"
: (i == 1 ? "Y" : "Z")) << ": ";
            if (cin >> input_speed[i]) break;
            cout << "Input tidak valid. Harap masukkan angka." <<
endl;

            cin.clear(); // Bersihkan flag kesalahan
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); //
Buang input yang tidak valid
        }
    }
}

void displayFormula() const {
    cout << BLUE << "Rumus Inverse Kinematics:" << RESET << endl;
    cout << "V_i = cos(theta_i) * x + sin(theta_i) * y + z" << endl;
    cout << "Di mana theta_i adalah sudut roda ke-i dalam derajat."
<< endl;
}

};

int main() {
    UserInterface ui;
    Kinematics kinematics; // Instance untuk pengelolaan kinematics
    int input_speed[3];
    char repeat;

    do {
        // Pesan sambutan
        ui.welcomeMessage();

        // Tampilkan rumus yang digunakan
        ui.displayFormula();

        // Tanyakan apakah pengguna ingin menghitung inverse kinematics
        char choice;
        while (true) {
            cout << "Apakah Anda ingin menghitung inverse kinematics?
(y/n): ";
            cin >> choice;

            if (choice == 'y' || choice == 'Y' || choice == 'n' || choice
== 'N') {
                break; // Keluar dari loop jika input valid
            } else {

```

```

        cout << "Input tidak valid! Silakan masukkan 'y' atau
'n'." << endl;
    }
}

if (choice == 'y' || choice == 'Y') {
    // Input kecepatan translasi X, Y, dan rotasi Z
    ui.getInput(input_speed);

    // Hitung kecepatan motor berdasarkan input kecepatan
    kinematics.calculateMotorSpeed(input_speed);

    // Tanyakan format penyimpanan
    string format;
    while (true) {
        cout << "Apakah Anda ingin menyimpan output sebagai Plain
Text atau JSON? (plain/json): ";
        cin >> format;

        if (format == "plain" || format == "json") {
            break; // Keluar dari loop jika format valid
        } else {
            cout << "Format tidak dikenali! Silakan masukkan
'plain' atau 'json'." << endl;
        }
    }

    // Meminta nama file untuk menyimpan data
    string filename;
    cout << "Masukkan nama file (default: motor_output): ";
    cin.ignore(); // Bersihkan input buffer
    getline(cin, filename);

    if (filename.empty()) {
        filename = (format == "json") ? "motor_output.json" :
"motor_output.txt"; // Gunakan nama default jika tidak ada input
    } else {
        filename += (format == "json") ? ".json" : ".txt"; //
Tambahkan ekstensi sesuai format
    }

    // Simpan input dan hasil ke file
    kinematics.saveDataToFile(input_speed, format);

    // Tampilkan hasil kecepatan motor
    kinematics.displayMotorSpeeds();
} else {

```

```

        // Jika tidak ingin menghitung, tawarkan opsi untuk membaca
file
        cout << "Apakah Anda ingin melihat hasil yang disimpan di
file? (y/n): ";
        cin >> choice;

        while (choice != 'y' && choice != 'Y' && choice != 'n' &&
choice != 'N') {
            cout << "Input tidak valid! Silakan masukkan 'y' atau
'n'." << endl;
            cout << "Apakah Anda ingin melihat hasil yang disimpan di
file? (y/n): ";
            cin >> choice;
        }

        if (choice == 'y' || choice == 'Y') {
            // Meminta nama file dari pengguna
            string filename;
            cout << "Masukkan nama file yang ingin dibaca (default:
motor_output.txt): ";
            cin.ignore(); // Bersihkan input buffer
            getline(cin, filename);

            if (filename.empty()) {
                filename = "motor_output.txt"; // Gunakan nama
default jika tidak ada input
            }

            kinematics.readDataFromFile(filename);
        }
    }

    // Tanyakan apakah pengguna ingin menghitung lagi
    cout << "Apakah Anda ingin menghitung lagi? (y/n): ";
    cin >> repeat;

    // Bersihkan input buffer
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    } while (repeat == 'y' || repeat == 'Y');

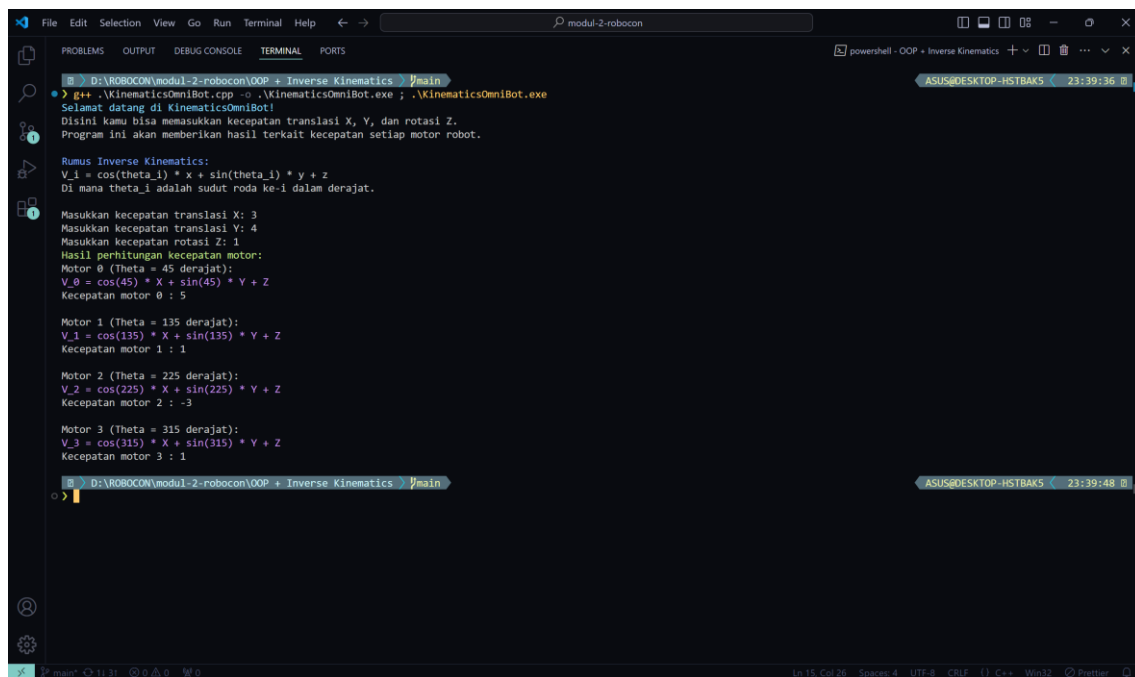
    return 0;
}

```

BAB III

OUTPUT PROGRAM

3.1 Program 1: KinematicsOmniBot



```
File Edit Selection View Go Run Terminal Help mod-2-robocon
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
D:\ROBOCON\modul-2-robocon\OOP + Inverse Kinematics /main
> g++ -lKinematicsOmniBot.cpp -o .\KinematicsOmniBot.exe ; .\KinematicsOmniBot.exe
Selamat datang di KinematicsOmniBot!
Disini kamu bisa memasukkan kecepatan translasi X, Y, dan rotasi Z.
Program ini akan memberikan hasil terkait kecepatan setiap motor robot.

Rumus Inverse Kinematics:
V_i = cos(theta_i) * x + sin(theta_i) * y + z
Di mana theta_i adalah sudut roda ke-i dalam derajat.

Masukkan kecepatan translasi X: 3
Masukkan kecepatan translasi Y: 4
Masukkan kecepatan rotasi Z: 1
Hasil perhitungan kecepatan motor:
Motor 0 (Theta = 45 derajat):
V_0 = cos(45) * X + sin(45) * Y + Z
Kecepatan motor 0 : 5

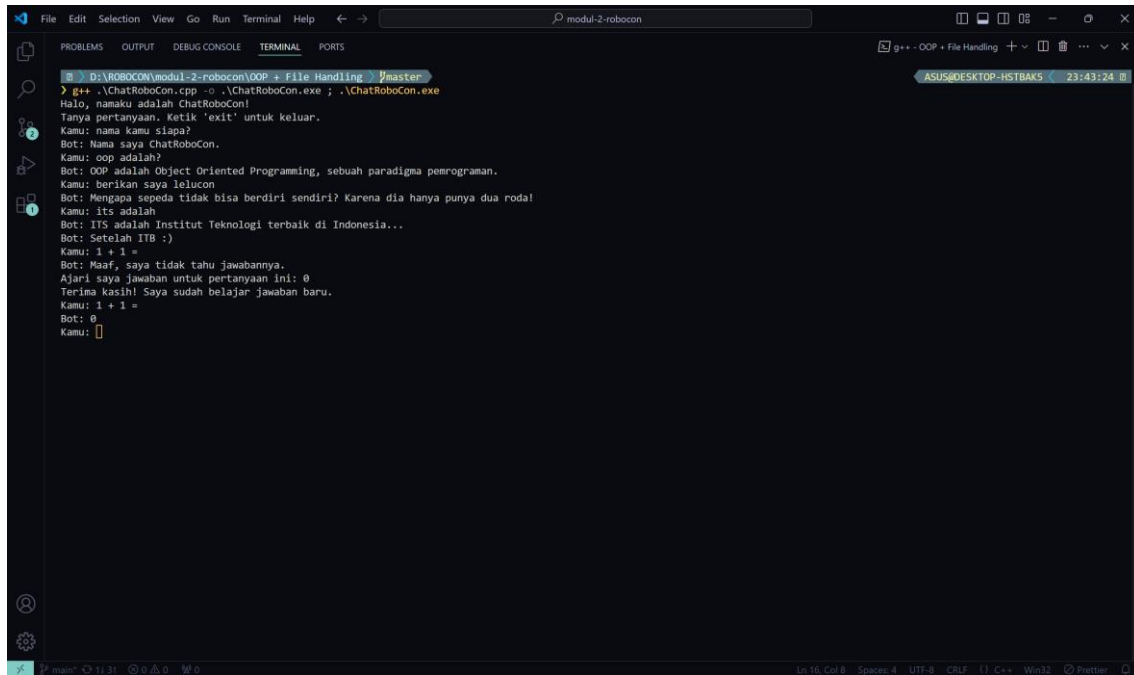
Motor 1 (Theta = 135 derajat):
V_1 = cos(135) * X + sin(135) * Y + Z
Kecepatan motor 1 : 1

Motor 2 (Theta = 225 derajat):
V_2 = cos(225) * X + sin(225) * Y + Z
Kecepatan motor 2 : -3

Motor 3 (Theta = 315 derajat):
V_3 = cos(315) * X + sin(315) * Y + Z
Kecepatan motor 3 : 1

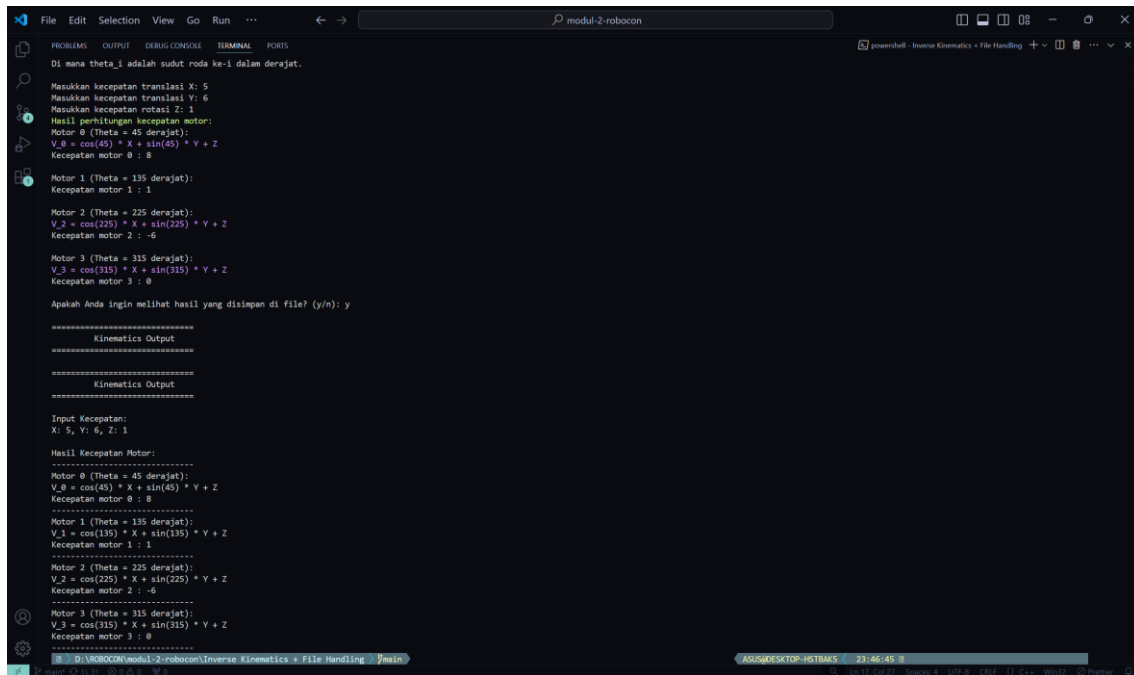
D:\ROBOCON\modul-2-robocon\OOP + Inverse Kinematics /main
>
```

3.2 Program 2: Question Answering Bot (ChatRoboCon)



```
File Edit Selection View Go Run Terminal Help ← → modul-2-robocon
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
D:\ROBOCON\modul-2-robocon\OOP + File Handling /master
g++ -o ChatRoboCon.exe .\ChatRoboCon.cpp & .\ChatRoboCon.exe
Halo, namaku adalah ChatRoboCon!
Tanya pertanyaan. Ketik 'exit' untuk keluar.
Kamu: nama kamu siapa?
Bot: Nama saya ChatRoboCon.
Kamu: oop adalah?
Bot: OOP adalah Object Oriented Programming, sebuah paradigma pemrograman.
Kamu: berikan saya lelucon
Bot: Mengapa sepeda tidak bisa berdiri sendiri? Karena dia hanya punya dua roda!
Kamu: its adalah
Bot: ITS adalah Institut Teknologi terbaik di Indonesia...
Bot: Setelah ITB :)
Kamu: 1 + 1 =
Bot: Maaf, saya tidak tahu jawabannya.
Ajari saya jawaban untuk pertanyaan ini: 0
Terima Kasih! Saya sudah belajar jawaban baru.
Kamu: 1 + 1 =
Bot: 0
Kamu: []
```

3.3 Program 3: Kinematics Data Logger (KinematicsAnalytics)



```
File Edit Selection View Go Run ... modul-2-robocon
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Di mana theta_i adalah sudut roda ke-i dalam derajat.
Masukkan kecepatan translasi X: 5
Masukkan kecepatan translasi Y: 6
Masukkan kecepatan rotasi Z: 1
Hasil perhitungan kecepatan motor:
Motor 0 (Theta = 45 derajat):
V_0 = cos(45) * X + sin(45) * Y + Z
Kecepatan motor 0 : 8
Motor 1 (Theta = 135 derajat):
Kecepatan motor 1 : 1
Motor 2 (Theta = 225 derajat):
V_2 = cos(225) * X + sin(225) * Y + Z
Kecepatan motor 2 : -6
Motor 3 (Theta = 315 derajat):
V_3 = cos(315) * X + sin(315) * Y + Z
Kecepatan motor 3 : 0
Apakah Anda ingin melihat hasil yang disimpan di file? (y/n): y
=====
Kinematics Output
=====
Kinematics Output
=====
Input Kecepatan:
X: 5, Y: 6, Z: 1
Hasil Kecepatan Motor:
=====
Motor 0 (Theta = 45 derajat):
V_0 = cos(45) * X + sin(45) * Y + Z
Kecepatan motor 0 : 8
=====
Motor 1 (Theta = 135 derajat):
V_1 = cos(135) * X + sin(135) * Y + Z
Kecepatan motor 1 : 1
=====
Motor 2 (Theta = 225 derajat):
V_2 = cos(225) * X + sin(225) * Y + Z
Kecepatan motor 2 : -6
=====
Motor 3 (Theta = 315 derajat):
V_3 = cos(315) * X + sin(315) * Y + Z
Kecepatan motor 3 : 0
=====
```

3.4 Program 4: Enhanced Kinematics (modul2robocon)

```
File Edit Selection View Go Run ... modul-2-robocon
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
D:\MSBOKCH\modul-2-robocon\OOP - Inverse Kinematics + File Handling >main
g++ -xmodul2robocon.cpp -o .\modul2robocon.exe ; .\modul2robocon.exe
Selamat datang di KinematicsOmiBot!
Disini kamu bisa memasukkan kecepatan translasi X, Y, dan rotasi Z.
Program ini akan memberikan hasil terkait kecepatan setiap motor robot.

Rumus Inverse Kinematics:
V_1 = cos(Theta_1) * X + sin(Theta_1) * Y + Z
Di mana Theta_1 adalah sudut roda ke-1 dalam derajat.
Apakah Anda ingin menghitung inverse kinematics? (y/n): y
Masukkan kecepatan translasi X: 4
Masukkan kecepatan translasi Y: 5
Masukkan kecepatan translasi Z: 1
Apakah Anda ingin menyimpan output sebagai Plain Text atau JSON? (plain/json): plain
Format tidak dikenali! Silakan masukkan 'plain' atau 'json'.
Apakah Anda ingin menyimpan output sebagai Plain Text atau JSON? (plain/json): plain
Masukkan nama file (default: motor_output):
Hasil perhitungan kecepatan motor:
Motor 0 (Theta = 45 derajat):
V_0 = cos(45) * X + sin(45) * Y + Z
Kecepatan motor 0 : 7

Motor 1 (Theta = 135 derajat):
V_1 = cos(135) * X + sin(135) * Y + Z
Kecepatan motor 1 : 1

Motor 2 (Theta = 225 derajat):
V_2 = cos(225) * X + sin(225) * Y + Z
Kecepatan motor 2 : -5

Motor 3 (Theta = 315 derajat):
V_3 = cos(315) * X + sin(315) * Y + Z
Kecepatan motor 3 : 0

Apakah Anda ingin menghitung lagi? (y/n): n
D:\MSBOKCH\modul-2-robocon\OOP - Inverse Kinematics + File Handling >main
>
```