LAPORAN PRAKTIKUM
KONSTRUKSI PERANGKAT BERGERAK

MODUL II

AUTOMATA DAN TABLE-DRIVEN CONSTRUCTION

Disusun Oleh :

Arwin Nabiel Arrofif

2211104057

S1SE-06-02

Asisten Praktikum :

Muhamad Taufiq Hidayat

Dosen Pengampu :

Riyan Dwi Yulian Prakoso, S.Kom., M.Kom.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

DIREKTORAT TELKOM KAMPUS PURWOKERTO

2025

BAB I
PENDAHULUAN

A. DASAR TEORI

Automata Construction

Automata Construction atau Automata-based programming adalah salah satu paradigma pemrograman dimana program dianggap seperti finite-state machine(FSM) atau formal automaton lainnya yang memiliki berbagai state-state yang saling berkaitan dan memiliki aturan tertentu yang jelas. Berikut ini indikator utama dalam Automata-based programming:

1. Jangka waktu eksekusi program dipisahkan dengan jelas pada state yang ada dan tidak terjadinya eksekusi yang overlaping pada state state yang ada.
2. Semua komunikasi antara state-state yang ada (perpindahan antar state) hanya dapat dilakukan secara eksplisit yang disimpan pada suatu global variable.

B. MAKSUD DAN TUJUAN

Tujuan dari praktikum ini adalah:

1. Memahami konsep Automata-Based Programming dan Table-Driven Construction.
2. Menerapkan Automata-Based Programming menggunakan bahasa C#.Net.
3. Menerapkan Table-Driven Construction dalam pengelolaan data dan kondisi program.
4. Membandingkan efisiensi penggunaan logika berbasis kondisi dengan metode berbasis tabel.

BAB II
IMPLEMENTASI (GUIDED)

Alamak.js

```javascript
const readline = require("readline");

class StateMachine {
    constructor() {
        this.states = {
            START: "START",
            GAME: "GAME",
            PAUSE: "PAUSE",
            EXIT: "EXIT"
        };
        this.currentState = this.states.START;

        this.rl = readline.createInterface({
            input: process.stdin,
            output: process.stdout
        });
    }

    run() {
        console.log(`\n${this.currentState} SCREEN`);
        this.askCommand();
    }

    askCommand() {
        this.rl.question("Enter Command: ", (command) => {
            this.transition(command.trim().toUpperCase());

            if (this.currentState === this.states.EXIT) {
                console.log("EXIT SCREEN");
                this.rl.close();
            } else {
                console.log(`\n${this.currentState} SCREEN`);
                this.askCommand();
            }
        });
    }

    transition(command) {
        switch (this.currentState) {
            case this.states.START:
                if (command === "ENTER") this.currentState = this.states.GAME;
                else if (command === "QUIT") this.currentState = this.states.EXIT;
                else console.log("Invalid command! Use ENTER to start or QUIT to exit.");
                break;

            case this.states.GAME:
                if (command === "ESC") this.currentState = this.states.PAUSE;
                else console.log("Unknown command! Press ESC to pause.");
                break;

            case this.states.PAUSE:
                if (command === "BACK") this.currentState = this.states.GAME;
                else if (command === "HOME") this.currentState = this.states.START;
                else if (command === "QUIT") this.currentState = this.states.EXIT;
                else console.log("Invalid command! Use BACK to resume, HOME to restart, or QUIT to
exit.");
                break;
        }
    }
}

// Jalankan State Machine
const game = new StateMachine();
game.run();
```

Output:

```
PS E:\Semester 6\kpl praktukum\pertemuan2> node alamak.js

START SCREEN
Enter Command: ENTER

GAME SCREEN
Enter Command: ESC

PAUSE SCREEN
Enter Command: BACK

GAME SCREEN
Enter Command: HOME
Unknown command! Press ESC to pause.

GAME SCREEN
Enter Command: ESC

PAUSE SCREEN
Enter Command: HOME

START SCREEN
Enter Command: ENTER

GAME SCREEN
Enter Command: QUIT
Unknown command! Press ESC to pause.

GAME SCREEN
Enter Command: EXIT
Unknown command! Press ESC to pause.

GAME SCREEN
Enter Command: ESC

PAUSE SCREEN
Enter Command: QUIT
EXIT SCREEN
PS E:\Semester 6\kpl praktukum\pertemuan2>
```

Welcome | alamak.js | node.js

```javascript
function getGradeByScore(studentScore) {
    const grades = ["A", "AB", "B", "BC", "C", "D", "E"];
    const rangeLimits = [80.0, 70.0, 65.0, 60.0, 50.0, 40.0, 0.0];
    let studentGrade = "E";
    let gradeLevel = 0;

    while (studentGrade === "E" && gradeLevel < grades.length - 1) {
        if (studentScore > rangeLimits[gradeLevel]) {
            studentGrade = grades[gradeLevel];
        }
        gradeLevel++;
    }

    return studentGrade;
}

// Contoh penggunaan
console.log(getGradeByScore(85)); // Output: "A"
console.log(getGradeByScore(72)); // Output: "AB"
console.log(getGradeByScore(50)); // Output: "C"
console.log(getGradeByScore(30)); // Output: "E"
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS E:\Semester 6\kpl praktukum\pertemuan2> node node.js
A
AB
D
E
PS E:\Semester 6\kpl praktukum\pertemuan2>
```

Ln 19, Col 35 Spaces: 4 UTF-8 CRLF {} JavaScript Go Live

BAB III
PENUGASAN (UNGUIDED)

Game.js

```js
const readline = require("readline");

class GameFSM {
    constructor() {
        this.states = {
            START: "START",
            PLAYING: "PLAYING",
            GAME_OVER: "GAME_OVER",
            EXIT: "EXIT"
        };
        this.currentState = this.states.START;

        this.rl = readline.createInterface({
            input: process.stdin,
            output: process.stdout
        });
    }

    run() {
        console.log(`\n${this.currentState} SCREEN`);
        this.askCommand();
    }

    askCommand() {
        this.rl.question("Enter Command: ", (command) => {
            this.transition(command.trim().toUpperCase());

            if (this.currentState === this.states.EXIT) {
                console.log("Exiting game...");
                this.rl.close();
            } else {
                console.log(`\n${this.currentState} SCREEN`);
                this.askCommand();
            }
        });
    }

    transition(command) {
        switch (this.currentState) {
            case this.states.START:
                if (command === "PLAY") this.currentState = this.states.PLAYING;
                else if (command === "EXIT") this.currentState = this.states.EXIT;
                else console.log("Invalid command! Use PLAY to start or EXIT to quit.");
                break;

            case this.states.PLAYING:
                if (command === "LOSE") this.currentState = this.states.GAME_OVER;
                else if (command === "EXIT") this.currentState = this.states.EXIT;
                else console.log("Invalid command! Use LOSE to end the game or EXIT to quit.");
                break;

            case this.states.GAME_OVER:
                if (command === "RESTART") this.currentState = this.states.START;
                else if (command === "EXIT") this.currentState = this.states.EXIT;
                else console.log("Invalid command! Use RESTART to play again or EXIT to quit.");
                break;
        }
    }
}

// Jalankan Game FSM
const game = new GameFSM();
game.run();
```

Output:

```
PS E:\Semester 6\kpl praktukum\pertemuan2> node game.js

START SCREEN
Enter Command: PLAY

PLAYING SCREEN
Enter Command: EXIT
Exiting game...
PS E:\Semester 6\kpl praktukum\pertemuan2> node game.js

START SCREEN
Enter Command: PLAY

PLAYING SCREEN
Enter Command: LOSE

GAME_OVER SCREEN
Enter Command: RESTART

START SCREEN
Enter Command: LOSE
Invalid command! Use PLAY to start or EXIT to quit.

START SCREEN
Enter Command:
```