

**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 overlapping 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

```
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();
      }
    });
  }

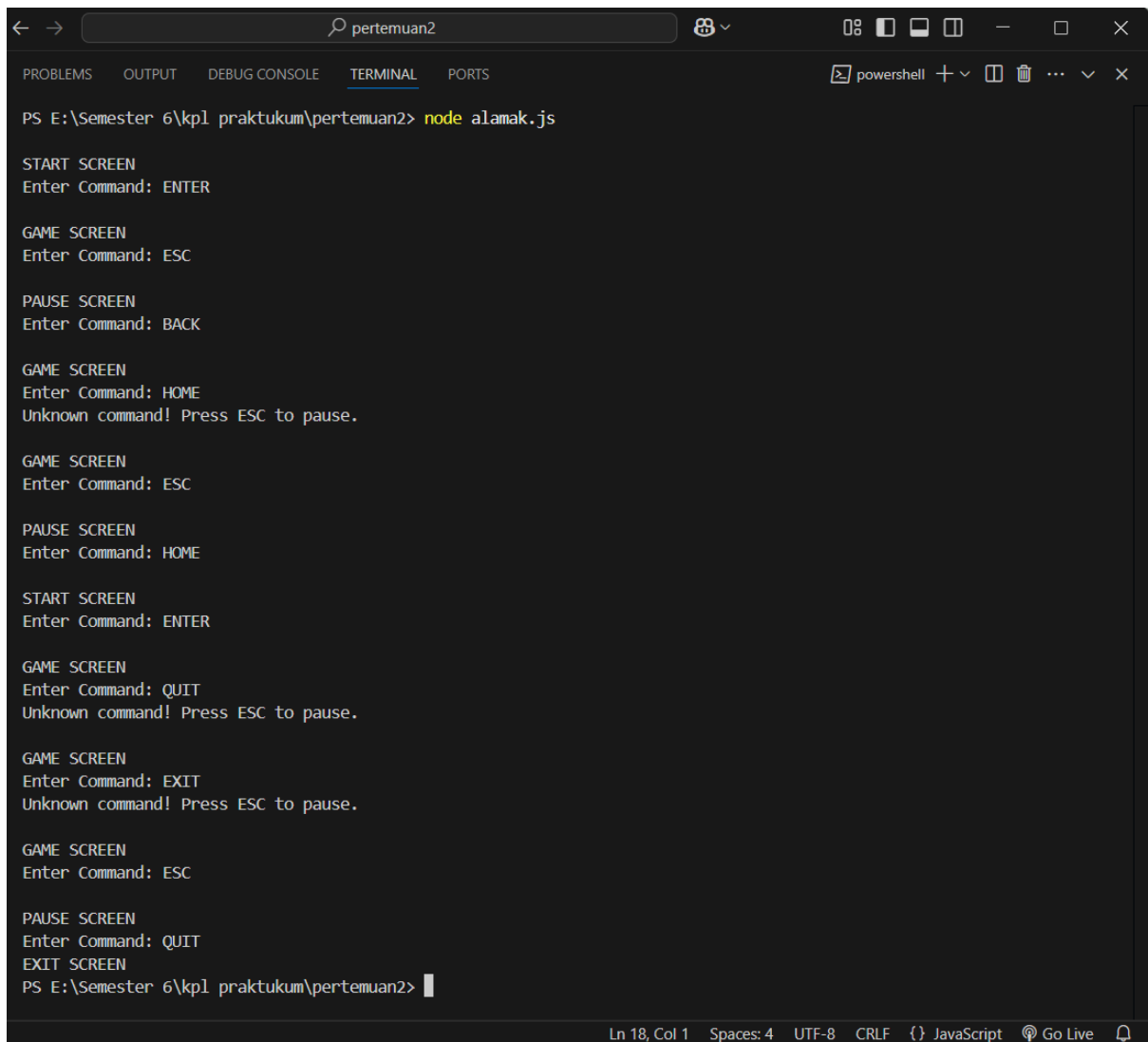
  transition(command) {
    switch (this.currentState) {
      case this.states.START:
        if (command === "ENTER") this.currentState = this.states.GAME;
        else if (command === "QUIT") this.currentState = this.states.EXIT;
        else console.log("Invalid command! Use ENTER to start or QUIT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



The screenshot shows a Visual Studio Code window with a terminal pane open. The terminal title is "pertemuan2". The command prompt shows the user is in the directory "E:\Semester 6\kpl praktikum\pertemuan2" and has run the command "node alamak.js". The output of the script is as follows:

```
PS E:\Semester 6\kpl praktikum\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 praktikum\pertemuan2>
```

The status bar at the bottom of the terminal indicates the current cursor position is "Ln 18, Col 1", the file has "Spaces: 4", the encoding is "UTF-8", the line endings are "CRLF", and the file type is "JavaScript". There is also a "Go Live" button and a bell icon.

The image shows a Visual Studio Code editor window with a dark theme. The top bar shows three tabs: 'Welcome', 'JS alamak.js', and 'JS node.js'. The 'JS node.js' tab is active, displaying a JavaScript function `getGradeByScore` that takes a `studentScore` and returns a grade based on a range of scores. The function uses a `while` loop to increment the grade level until the score is within the range. Below the function, there are four `console.log` statements demonstrating the function's output for scores 85, 72, 50, and 30. The bottom panel shows the 'TERMINAL' tab with a PowerShell prompt where the command `node node.js` has been executed, resulting in the output: A, AB, D, E.

```
1 function getGradeByScore(studentScore) {
2     const grades = ["A", "AB", "B", "BC", "C", "D", "E"];
3     const rangeLimits = [80.0, 70.0, 65.0, 60.0, 50.0, 40.0, 0.0];
4     let studentGrade = "E";
5     let gradeLevel = 0;
6
7     while (studentGrade === "E" && gradeLevel < grades.length - 1) {
8         if (studentScore > rangeLimits[gradeLevel]) {
9             studentGrade = grades[gradeLevel];
10        }
11        gradeLevel++;
12    }
13
14    return studentGrade;
15 }
16
17 // Contoh penggunaan
18 console.log(getGradeByScore(85)); // Output: "A"
19 console.log(getGradeByScore(72)); // Output: "AB"
20 console.log(getGradeByScore(50)); // Output: "C"
21 console.log(getGradeByScore(30)); // Output: "E"
22
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + - [ ] [ ] ... ^ x

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

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

### BAB III

## PENUGASAN (UNGUIDED)

Game.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:

```
JS game.js > ...
3  class GameFSM {

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
node + - [] ... ^ x

PS E:\Semester 6\kp1 praktikum\pertemuan2> node game.js

START SCREEN
Enter Command: PLAY

PLAYING SCREEN
Enter Command: EXIT
Exiting game...
PS E:\Semester 6\kp1 praktikum\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: 
```

Ln 64, Col 1 (2108 selected) Spaces: 4 UTF-8 CRLF {} JavaScript Go Live