Laporan Analisis Tugas Robotika Week 14

Nama Lengkap: Muhammad Nugraha Sadewa

Analisis Per File Kode

1. Arrays and Vectors

Deskripsi: Kode ini menunjukkan cara mendeklarasikan dan menggunakan array serta vektor di Rust. Array memiliki ukuran tetap, sedangkan vektor bersifat dinamis.

Fitur Utama:

- Deklarasi array dan akses elemen dengan indeks.
- Deklarasi vektor dengan nilai awal atau penambahan elemen menggunakan push.
- Penghapusan elemen vektor menggunakan pop.

Kode Penting:

```
let working_days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"];

println!("{}", working_days[0]);


let mut names = Vec::new();

names.push("Will");

names.pop();
```

Penjelasan:

- Elemen array diakses dengan indeks seperti working_days[0].
- Vec::new() membuat vektor kosong, dan push menambahkan elemen.

2. Data Types

Deskripsi: Kode ini menjelaskan tipe data bawaan Rust seperti integer, float, boolean, char, dan tuple. Struct klasik dan tuple struct juga diperkenalkan.

Fitur Utama:

- Deklarasi tipe data dengan eksplisit atau inferensi.
- Implementasi struct klasik dan tuple struct.

Kode Penting:

```
struct Student {
```

```
    name: String,
    level: u8,
    remote: bool,
}

let student_1 = Student {
    name: String::from("Nugraha"),
    level: 5,
    remote: true,
};
println!("{} is level {}", student_1.name, student_1.level);
```

Penjelasan:

- Struct klasik memungkinkan akses field menggunakan nama.

3. Event-Driven Programming

Deskripsi: Menggunakan kanal (mpsc) untuk komunikasi antar-thread dalam sistem berbasis event.

Fitur Utama:

- Thread untuk mengirim event seperti ObstacleDetected.
- Event diterima dengan kanal dan ditangani dalam loop utama.

Kode Penting:

```
let (tx, rx) = mpsc::channel();
thread::spawn(move || {
    tx.send(Event::ObstacleDetected).unwrap();
});
```

Penjelasan:

- Kanal digunakan untuk komunikasi aman antara thread.

4. Hash Map

Deskripsi: Penggunaan hash map untuk menyimpan pasangan key-value dan manipulasi datanya.

Fitur Utama:

- Menambahkan data dengan insert.
- Mengakses data dengan get.
- Menghapus data dengan remove.

Kode Penting:

```
let mut items = HashMap::new();

items.insert(String::from("One"), String::from("Book"));

println!("{:?}", items.get("One"));
```

Penjelasan:

- Hash map menyimpan data dengan key unik dan memberikan akses cepat ke value.

5. Hello Cargo

Deskripsi: Program sederhana untuk memperkenalkan struktur proyek Rust dengan Cargo.

Kode Penting:

```
fn main() {
   println!("Hello, world!");
}
```

Penjelasan:

- Program dasar untuk memastikan Cargo terinstal dan berfungsi.

6. Hello World

Deskripsi: Program minimalis yang mencetak "Hello, World!" ke terminal.

Kode Penting:

```
fn main() {
   println!("Hello, World!");
}
```

Penjelasan:

- Makro println! mencetak teks ke terminal.

7. If-Else

Deskripsi: Penggunaan alur kontrol if-else untuk berbagai kondisi.

Fitur Utama:

- Menentukan alur berdasarkan kondisi boolean.
- Menggunakan if-else untuk inisialisasi variabel.

Kode Penting:

```
let take_jacket = if sunny_day { "Don't take a jacket" } else { "Take a jacket" };
```

Penjelasan:

- Kondisi if-else langsung menginisialisasi variabel.

8. Loops

Deskripsi: Berbagai jenis loop di Rust seperti loop, while, dan for.

Fitur Utama:

- Menggunakan break untuk keluar dari loop.
- Iterasi koleksi menggunakan for.

Kode Penting:

```
let loop_stop = loop {
  counter *= 4;
  if counter > 100 {
    break counter;
  }
};
```

Penjelasan:

- Nilai dapat dikembalikan saat keluar dari loop dengan break.

9. Prioritas

Deskripsi: Menggunakan BinaryHeap untuk implementasi antrean prioritas.

Fitur Utama:

- Menambahkan tugas dengan push.
- Mengambil tugas prioritas tertinggi dengan pop.

Kode Penting:

```
while let Some(task) = task_queue.pop() {
    println!("Menyelesaikan tugas: {}", task.description);
}
```

Penjelasan:

- Tugas diambil dalam urutan prioritas tertinggi.

10. Probabilistik

Deskripsi: Robot bergerak menuju tujuan dengan memperhitungkan ketidakpastian.

Fitur Utama:

- Menggunakan crate rand untuk simulasi noise.
- Menghitung posisi baru berdasarkan langkah dan noise.

Kode Penting:

```
let uncertainty_x: f64 = rng.gen_range(-0.5..0.5);
position.0 = ((position.0 as f64 + step_x as f64 + uncertainty_x).round() as i32).max(0) as usize;
```

Penjelasan:

- Noise ditambahkan ke posisi untuk mensimulasikan ketidakpastian.

11. Robot Finding (A*)

Deskripsi: Implementasi algoritma A* untuk menemukan jalur terpendek.

Fitur Utama:

- BinaryHeap digunakan untuk memilih node dengan biaya terendah.
- Heuristik Manhattan digunakan untuk estimasi jarak.

Kode Penting:

```
let mut open_set = BinaryHeap::new();
open_set.push(Node {
    point: start,
    g_cost: 0,
    h_cost: manhattan_distance(start, goal),
    parent: None,
```

});

Penjelasan:

- BinaryHeap membantu memilih node prioritas tertinggi (biaya rendah).

12. Robot Input

Deskripsi: Robot menerima input pengguna untuk bergerak ke arah tertentu.

Fitur Utama:

- Input diterima dari terminal dengan io::stdin.
- Gerakan diproses berdasarkan input.

Kode Penting:

```
match input.trim() {
  "w" => position.1 += 1,
  "a" => position.0 -= 1,
  _ => println!("Input invalid"),
}
```

Penjelasan:

- Input dipetakan ke perubahan posisi robot.

13. Sederhana (BFS Pathfinding)

Deskripsi: Menggunakan algoritma BFS untuk menemukan jalur dalam grid.

Fitur Utama:

- Implementasi BFS dengan antrian (VecDeque).
- Validasi langkah untuk memastikan posisi tidak keluar dari grid.

Kode Penting:

```
let directions = [(-1, 0), (1, 0), (0, -1), (0, 1)];
if is_valid_move(nx, ny, grid, &visited) {
  queue.push_back(next);
}
```

Penjelasan:

- Validasi posisi baru untuk memastikan langkah valid.

Kesimpulan

Laporan ini mencakup analisis kode dari berbagai tugas. Rust terbukti sangat kuat untuk aplikasi seperti pemrograman sistem, algoritma, dan robotika karena tipe data yang ketat, keamanan memori, dan kemampuan multithreading.