Nama : Ketut Satria Wibisana
NIM : 1103213148
Kelas : TK-45-G09

Analisis Learn Rust with Me! Part 1-6

1. **Hello-cargo**

   Kode ini merupakan contoh dasar program Rust yang digunakan untuk mencetak "Hello, world!" dengan memanfaatkan fungsi `main` sebagai titik awal eksekusi. Macro `println!` digunakan untuk menampilkan teks ke konsol, yang memperlihatkan sintaks bawaan Rust yang mendukung pencetakan langsung. Program ini menggambarkan struktur dasar aplikasi Rust yang biasanya dibuat dengan menggunakan Cargo, alat pengelola proyek dan paket Rust. Kode ini menunjukkan bagaimana Rust didesain untuk memulai pengembangan dengan cara yang sederhana namun efektif, mengikuti prinsip setup default.

2. **Data-types**

   Kode ini menggambarkan berbagai cara Rust menangani tipe data melalui penggunaan struct, tuple struct, variabel immutable dan mutable, serta berbagai tipe data dasar seperti integer, float, boolean, dan string. Struct konvensional memungkinkan pengelolaan data dengan field yang bernama, sementara tuple struct hanya mendefinisikan tipe data tanpa memberikan nama pada field. Rust juga mendukung shadowing, yang memungkinkan pembaruan nilai variabel secara aman tanpa menggantikan deklarasi awal. Dengan tipe data yang bersifat statis, Rust menjamin kejelasan dan keamanan tipe pada saat kompilasi. Kode ini menyoroti kemampuan Rust dalam menangani data secara fleksibel, baik dalam bentuk primitif maupun struktur yang lebih kompleks.

3. **If-else**

   Kode ini menunjukkan penggunaan pernyataan if-else dalam Rust yang tidak hanya memungkinkan evaluasi kondisi, tetapi juga dapat mengembalikan nilai. Ekspresivitas ini memudahkan penetapan nilai langsung ke variabel berdasarkan kondisi yang ada. Evaluasi kondisi berlapis menggunakan if-else if-else mencerminkan bagaimana Rust mendorong penulisan kode yang eksplisit dan terstruktur. Selain itu, Rust mengharuskan inisialisasi variabel sebelum digunakan, yang memastikan bahwa kode terhindar dari akses nilai yang tidak terdefinisi. Fitur ini menggambarkan pendekatan Rust yang mengutamakan keamanan dan keandalan dalam penulisan kode.

4. **Arrays-and-vectors**

   Kode ini menjelaskan bagaimana Rust menangani koleksi data melalui penggunaan array dan vektor. Array merupakan kumpulan data dengan ukuran tetap dan tipe yang seragam, sementara vektor memberikan fleksibilitas untuk menambah atau menghapus elemen. Operasi seperti push dan pop pada vektor menunjukkan bagaimana Rust memungkinkan manipulasi

data secara dinamis. Vektor dalam Rust memerlukan deklarasi mut agar dapat diubah, mencerminkan filosofi keamanan Rust yang membedakan antara data yang mutable dan immutable. Pendekatan ini mempermudah pengelolaan koleksi data dengan cara yang aman dan efisien.

5. **Loops**

   Kode ini memperlihatkan berbagai bentuk loop dalam Rust, termasuk loop tanpa henti, while loop, dan for loop. Loop tanpa henti dapat dihentikan menggunakan perintah break, bahkan dengan mengembalikan nilai pada titik tersebut. While loop memungkinkan iterasi selama kondisi tertentu terpenuhi, sementara for loop menggunakan iterator untuk memproses koleksi atau rentang nilai. Dengan kombinasi ini, Rust menawarkan fleksibilitas tinggi dalam pengulangan, memungkinkan pengembang untuk menangani berbagai skenario iterasi dengan sintaks yang sederhana namun jelas.

6. **Hash-map**

   Kode ini menyoroti penggunaan HashMap dalam Rust, sebuah koleksi pasangan kunci-nilai yang disediakan melalui modul std::collections. Operasi seperti insert untuk menambah pasangan kunci-nilai, get untuk mengambil nilai berdasarkan kunci, dan remove untuk menghapus kunci menggambarkan kemampuan dasar HashMap. Nilai yang diambil dengan get berupa Option, yang mencerminkan sistem tipe Rust yang aman dalam menangani kemungkinan ketidakadaan nilai. Dengan mendesain operasi hash map secara eksplisit dan aman, Rust memungkinkan pengelolaan data asosiatif secara efisien sekaligus mencegah bug yang umum terjadi pada koleksi kunci-nilai.

Analisis Project Rust

1. **Perencanaan Jalur Sederhana**

   Kode ini mengimplementasikan algoritma Breadth-First Search (BFS) untuk mencari jalur terpendek dari titik awal ke titik tujuan dalam grid 2D. Grid direpresentasikan sebagai matriks, di mana nilai 0 menunjukkan jalan yang dapat dilalui dan nilai 1 menunjukkan rintangan. Fungsi `is_valid_move` memeriksa validitas pergerakan dengan mempertimbangkan batas grid, kondisi jalan, dan status kunjungan. BFS dijalankan menggunakan `VecDeque` untuk mengelola antrian posisi yang perlu dijelajahi. Jalur ditampilkan sebagai urutan koordinat (x, y) yang diperbarui pada setiap langkah, dengan memastikan bahwa posisi yang sama tidak dilalui lebih dari sekali. Program ini menggambarkan pendekatan yang efisien dan aman dalam pencarian jalur.

2. **Gerakan Robot dengan Input Pengguna**

   Kode ini mensimulasikan gerakan robot di grid 2D berdasarkan input pengguna. Posisi robot

diperbarui sesuai dengan perintah yang dimasukkan (up, down, left, atau right). Validasi perintah
dilakukan menggunakan pola match, di mana perintah yang tidak dikenal akan menghasilkan pesan
kesalahan. Perintah exit digunakan untuk keluar dari program. Dengan loop tak terbatas, kode ini
terus menerima input dan mencetak posisi robot secara dinamis. Pendekatan ini efektif untuk simulasi robot berbasis perintah langsung, memungkinkan pengguna untuk mengontrol gerakan
secara real-time.

3. **Simulasi Robot Menghindari Rintangan**

   Kode ini mengembangkan contoh perencanaan jalur dengan menambahkan visualisasi jalur yang diambil oleh robot dalam grid yang berisi rintangan. Seperti pada contoh perencanaan jalur sederhana, algoritma BFS digunakan untuk menemukan jalur dari titik awal ke tujuan. Namun, pada kode ini, elemen visualisasi ditambahkan untuk menggambarkan grid dan langkah-langkah perjalanan robot, menunjukkan bagaimana robot menghindari rintangan dan melaporkan jalur yang diambil. Dengan mencetak setiap langkah yang diambil, kode ini memberikan gambaran lebih jelas mengenai proses eksplorasi robot dalam lingkungan yang kompleks.

4. **Penjadwalan Robot dengan Prioritas**

   Kode ini mensimulasikan penjadwalan tugas robot menggunakan struktur data BinaryHeap. Setiap tugas diberikan prioritas, di mana tugas dengan prioritas tertinggi akan dieksekusi terlebih dahulu. Prioritas ini diimplementasikan dengan membalik urutan heap menggunakan metode `cmp` pada struktur `Task`. Ketika tugas ditambahkan ke antrean, tugas-tugas dieksekusi sesuai urutan prioritas menggunakan operasi `pop`. Pendekatan ini sangat efisien untuk manajemen tugas dinamis, memastikan bahwa tugas-tugas yang lebih penting dieksekusi lebih awal, yang merupakan kebutuhan umum dalam sistem robotik.

5. **Robotik dengan Sistem Event-Driven**

   Kode ini mengimplementasikan paradigma event-driven untuk mengontrol perilaku robot dalam merespons berbagai kejadian, seperti mendeteksi rintangan atau perubahan target. Robot memiliki posisi dan target, serta dapat bergerak menuju target atau berhenti ketika menghadapi rintangan. Kejadian-kejadian tersebut disimulasikan menggunakan thread terpisah yang secara periodik menghasilkan peristiwa seperti `ObstacleDetected`, `TargetChanged`, atau `Idle`. Sinkronisasi antara thread dilakukan menggunakan `Mutex` untuk memastikan akses aman ke data bersama. Pendekatan ini meniru sistem robotik dunia nyata yang merespons lingkungan secara asinkron dan responsif.

6. **Robot dengan Model Probabilistik**

   Kode ini mengimplementasikan pendekatan particle filter untuk memperkirakan posisi robot dalam lingkungan yang mengandung ketidakpastian. Setiap partikel mewakili

kemungkinan posisi robot, dengan posisi dan orientasi yang diperbarui berdasarkan pergerakan dan data sensor. Bobot partikel dihitung berdasarkan kesesuaian antara data sensor dan prediksi model lingkungan. Setelah pembaruan bobot, dilakukan resampling untuk memilih partikel-partikel yang lebih sesuai. Proses ini meningkatkan akurasi estimasi posisi robot seiring berjalannya waktu. Dengan simulasi data sensor dan peta sederhana, kode ini menunjukkan bagaimana algoritma probabilistik dapat digunakan untuk navigasi robotik dalam lingkungan yang tidak pasti.