Hendri Maulana Azwar

1103210202

TK-45-G09

Week 14

Tutor 1 :

Video ini menjelaskan langkah-langkah untuk menginstal **Rust** di komputer dan menunjukkan cara membuat program pertama, yaitu **Hello World**. Langkah pertama adalah mengunduh dan menginstal Rust melalui rustup, yang mengatur pengelolaan versi Rust. Setelah instalasi selesai, video menunjukkan cara membuat file sumber Rust, menulis program "Hello, World!", dan menjalankannya menggunakan perintah cargo run. Ini memberikan pemahaman dasar bagi pemula tentang cara mulai bekerja dengan Rust, termasuk pengaturan lingkungan pengembangan dan eksekusi program pertama.

Tutor 2 :

Dari video terdapat Kode yang mendemonstrasikan konsep dasar Rust, termasuk penggunaan **struct** dan **tuple** untuk mengorganisasi data, variabel mutable dan shadowing untuk memperbarui nilai, serta operasi logika dan aritmatika sederhana. **Struct Student** menyimpan informasi siswa, sementara **tuple Grades** menyimpan nilai akademik. Kode ini juga menampilkan penggunaan println! untuk output dengan format string dan contoh pengolahan data sederhana, seperti perhitungan usia dan pencetakan atribut dari struct dan tuple. Program ini cocok sebagai latihan pemahaman dasar Rust secara terstruktur dan fungsional.

Tutor 3 :

Dari video terdapat Kode yang menunjukkan penggunaan **if/else statement** dalam Rust untuk mengevaluasi kondisi logis. Program ini mencakup pengambilan keputusan sederhana dengan mencetak pernyataan apakah angka sama atau tidak, menentukan apakah perlu membawa jaket berdasarkan kondisi cuaca, dan mengevaluasi apakah sebuah angka berada di luar rentang tertentu menggunakan beberapa kondisi if/else. Kode ini memperlihatkan cara mengikat nilai ke variabel menggunakan ekspresi if/else dan menunjukkan fleksibilitas Rust dalam menangani logika kompleks secara terstruktur.

Tutor 4 :

Dari video terdapat Kode yang mendemonstrasikan penggunaan **array** dan **vector** di Rust, yang digunakan untuk menyimpan koleksi data dengan tipe yang sama. Program ini menunjukkan cara mendeklarasikan array dengan nilai inisialisasi dan panjang tetap, serta cara mengakses elemen array berdasarkan indeks. Untuk vector, kode ini memperlihatkan cara mendeklarasikan, menambahkan (menggunakan push), menghapus (pop), dan memodifikasi elemen vector. Selain itu, vector memungkinkan fleksibilitas dalam ukuran dibandingkan array.

Program ini menggambarkan bagaimana Rust mengelola koleksi data secara efisien dengan berbagai metode manipulasi.

Tutor 5 :

Dari video terdapat Kode yang menunjukkan penggunaan berbagai jenis **loop** di Rust: **loop**, **while loop**, dan **for loop**. Program ini memulai dengan loop yang berjalan selamanya (meskipun dinonaktifkan dengan komentar), dan kemudian memperlihatkan penggunaan break untuk keluar dari loop dan mengembalikan nilai. **While loop** digunakan untuk menjalankan suatu aksi selama kondisi tertentu masih terpenuhi, sementara **for loop** menggunakan iterator untuk memproses elemen-elemen dalam koleksi atau rentang angka. Program ini juga menunjukkan cara menggunakan metode iter() untuk mengakses elemen dalam array dan cara menggunakan rentang (0..10) untuk iterasi angka. Rust memberikan fleksibilitas dalam memilih tipe loop yang sesuai dengan kebutuhan pemrograman.

Tutor 6 :

Dari video terdapat Kode yang menunjukkan penggunaan **HashMap** di Rust, yang merupakan struktur data untuk menyimpan pasangan **key-value**. Program ini mengimpor **HashMap** dari pustaka standar Rust dan mengilustrasikan cara membuat **HashMap** kosong menggunakan HashMap::new(). Setelah itu, elemen dimasukkan ke dalam map menggunakan metode insert(<key>, <value>). Untuk mengambil nilai berdasarkan kunci, digunakan metode get(<key>). Program ini juga menunjukkan cara menghapus elemen dari map dengan metode remove(<key>) dan bagaimana cara menangani nilai yang telah dihapus dengan mengembalikan None. Ini memberikan gambaran tentang cara mengelola data secara efisien dengan menggunakan HashMap dalam Rust.

Pembuatan kode percobaan robot :

Kode tersebut menggambarkan simulasi **robot** yang dapat menerima berbagai perintah untuk bergerak dan berputar menggunakan struktur **enum** dan **struct** di Rust. **Enum Command** mendefinisikan berbagai perintah yang bisa diberikan pada robot, seperti maju, mundur, belok kiri, belok kanan, dan berhenti. **Struct Robot** menyimpan posisi (x, y) dan arah robot. Program ini menggunakan metode execute_command untuk menjalankan perintah yang diberikan, dengan menggunakan pola **match** untuk memproses perintah berdasarkan tipe enum. Program ini meminta input dari pengguna melalui stdin, memungkinkan interaksi untuk menggerakkan robot berdasarkan perintah yang dimasukkan. Setelah perintah berhenti diterima, simulasi akan berakhir. Kode ini memberikan gambaran praktis tentang penggunaan enum, struct, dan input/output dalam Rust.