Nama : Eko Putra Nugraha

NIM : 1103213212

1.Struktur dan Dasar Program

Fragmen kode [11] menunjukkan implementasi dasar program Rust dengan fungsi main() sederhana yang hanya mencetak "Hello, world!". Ini merupakan program dasar dalam Rust, digunakan untuk menguji lingkungan dan konfigurasi kompilator.

2. Manipulasi Data dan Koleksi

Kode di [12] menampilkan manipulasi array, vektor, dan pengolahan data dalam Rust:

- Array statis seperti working_days digunakan untuk menyimpan daftar hari kerja.
- Vektor dinamis seperti nephews_age dan names menunjukkan fleksibilitas Rust dalam menangani ukuran data yang berubah-ubah. Operasi seperti push dan pop digunakan untuk menambahkan atau menghapus elemen.
- Kode ini juga menampilkan bagaimana elemen dapat diakses, dimodifikasi, dan dicetak.

3. Penggunaan Struktur dan Tipe Data

Pada [13], penggunaan struct (Student dan Grades) menunjukkan kemampuan Rust untuk mendefinisikan tipe data kompleks:

- Struct digunakan untuk merepresentasikan data dengan berbagai properti, seperti nama, level, dan status kerja jarak jauh.
- Program juga mencakup operasi sederhana dengan tipe primitif seperti integer, boolean, float, dan string. Contohnya adalah pengolahan umur dan penggunaan tuple untuk informasi hewan peliharaan.

4. Pemanfaatan HashMap

Di [14], Rust menunjukkan dukungan untuk koleksi asosiatif melalui HashMap:

- Elemen-elemen dimasukkan dengan kunci dan nilai (contohnya: "One" -> "Book").
- Operasi seperti insert, get, dan remove ditunjukkan untuk manipulasi data.

5. Pengambilan Keputusan dan Logika

Kode di [16] menampilkan implementasi pengambilan keputusan menggunakan if-else:

- Contoh kondisi sederhana seperti membandingkan angka (1 == 2) digunakan untuk mendemonstrasikan alur logika.
- Struktur bersyarat yang kompleks digunakan untuk menentukan rentang nilai (out_of_range).

6. Perulangan

Pada [17], Rust menampilkan berbagai mekanisme perulangan:

- Perulangan dengan while untuk mencetak pesan sejumlah iterasi.

- Perulangan for digunakan untuk iterasi array (shopping_list) dan rentang angka (0..10).
- Contoh perulangan loop diakomodasi tetapi dikomentari, yang mengindikasikan pengujian perulangan tanpa akhir.

Prompt
Implementasi Algoritma Pencarian Jalur (Pathfinding)

Kode pada [27] dan [29] mengimplementasikan algoritma pencarian jalur menggunakan pendekatan *A (A-Star)** dan **Breadth-First Search (BFS)**:

- **A***: Memanfaatkan BinaryHeap untuk menjaga antrian prioritas berdasarkan heuristic (Manhattan distance) dan biaya (cost). Implementasi ini sangat efisien untuk mencari jalur optimal dalam grid dengan hambatan.
- **BFS**: Menggunakan VecDeque sebagai struktur antrian. BFS memastikan jalur yang ditemukan adalah yang terpendek, meskipun tidak optimal dalam hal biaya karena tidak menggunakan heuristic.

Kedua pendekatan ini cocok untuk aplikasi navigasi robot atau gim berbasis grid.

---

2. Manajemen Tugas dengan Prioritas

Pada [28], kode memperkenalkan konsep antrian prioritas menggunakan BinaryHeap untuk menyusun tugas berdasarkan prioritas:

- Tugas dengan prioritas tertinggi diproses lebih dahulu.
- Struktur ini berguna dalam skenario seperti penjadwalan tugas robot atau manajemen proses dalam sistem operasi.

---

3. Navigasi dan Interaksi dengan Pengguna

Kode pada [30] menyediakan antarmuka berbasis teks untuk mengontrol pergerakan posisi dalam grid:

- Input pengguna (w/a/s/d) memungkinkan navigasi dua dimensi.
- Kode ini menekankan bagaimana loop dapat digunakan untuk interaksi berulang hingga perintah keluar diberikan (q).

---

4. Simulasi Robot dengan Ketidakpastian

Di [31], robot virtual mensimulasikan navigasi dengan ketidakpastian dalam sensor dan gerakan:

- Robot mencoba mencapai target dengan pengaruh noise acak pada posisi dan gerakannya.
- Implementasi ini sangat relevan untuk robotika dunia nyata, di mana ketidakpastian adalah hal yang umum, seperti dalam **Monte Carlo Localization**.

---

5. Penggunaan Struktur Data Kompleks

Kode menggunakan berbagai struktur data standar Rust, seperti:

- **HashMap** untuk memetakan hubungan (contohnya: item dan deskripsi tugas di [28]).
- **VecDeque** untuk BFS di [29].
- **BinaryHeap** untuk antrian prioritas di [28].

Pemanfaatan struktur ini menunjukkan fleksibilitas Rust dalam membangun solusi kompleks dengan pendekatan yang efisien.