Makra deklaratywne i typy zaawansowane

19 grudnia 2025

Instrukcja

Zadanie zaimplementuj jako bibliotekę. Dopuszczalne jest użycie tylko biblioteki standardowej.

1. Zaimplementuj wszystkie elementy z sekcji *Funkcjonalność*.
2. Napraw wszystkie ostrzeżenia kompilatora.
3. Sprawdź, czy program przechodzi testy (`cargo test`).
4. Użyj `cargo clippy`, aby znaleźć typowe problemy i je poprawić.

Funkcjonalność

1. Zaimplementuj makro `string!`, które przyjmowany argument (w założeniu `&str`) zamienia na `String` używając `String::from` lub `.to_string()`.
2. Zdefiniuj trait `StateMachine<S>` (automat skończony) z metodą

   ```
   fn step(&self, state: S) -> Option<S>;
   ```

   Zwrócenie przez `step` wartości `None` oznaczać będzie, że automat skończył pracę.
3. Zaimplementuj makro `impl_state_machine`, które definiuje strukturę i implementację `StateMachine<i32>` dla niej. Można je wywołać, przykładowo, w następujący sposób:

   ```
   impl_state_machine!(MyMachine, [
   1 -> 3;
   2 -> 3;
   3 -> 4;
   4 -> END;
   ])
   ```

   Gdzie `MyMachine` to nazwa definiowanej struktury, `i -> j`, to przejście ze stanu i do stanu j, a `END` to stan końcowy.

   Podpowiedź: najłatwiej w `MyMachine` zawrzeć `HashMap`, i `i -> j` mapować na `map.insert(i, j)`, a przed rozwinięciem powtórzenia w makrze zdefiniować `const END: i32 = -1`. Jeśli masz dużo czasu, możesz spróbować bez użycia mapy, z samym `match`.
4. Dla każdego typu `S` implementującego `Clone`, `Eq` i `Hash` zaimplementuj `StateMachine<S>` dla każdej `HashMap<S, S>`, traktując mapę jak tablicę przejść.
5. Napisz funkcję generyczną dla `M1`, `M2` implementujących `StateMachine<S>` dla tego samego S, zastępując `?` odpowiednimi zapisami

```
fn join_machines<?, ?, M1, M2>(x: M1, y: M2) -> ?
where
?
{
    vec![Box::new(x), Box::new(y)]
}
```

Podpowiedź: pamiętaj o okresach życia!

6. Użyj zdefiniowanych przez siebie makr, funkcji i struktur.

Wymagania

- Kod kompiluje się ze stabilnym kompilatorem Rust.
- Brak ostrzeżeń kompilatora i `clippy` (w tym o nieużywanych elementach).
- Wszystkie testy przechodzą.

Ocena (3 pkt)

- 1 pkt: Praca w trakcie laboratorium.
- 1 pkt: Pełna funkcjonalność (implementacja zgodna z wymaganiami).
- 1 pkt: Prezentacja rozwiązania i odpowiedzi na pytania prowadzącego.