RUST PROGRAMMING

ITERATORS

Kamal kumar mukiri,
Apt Computing Labs

Use of Iterators

Problems with “loops”:

Increased Risk of Errors: Manual loops require careful handling of index bounds

Lower Performance: Without iterators, manual loops might not benefit as much from compiler optimizations

Reduced Code Clarity: Code written with manual loops can become verbose and harder to read, especially as complexity grows.

How does Rust Compiler optimize “ITERATORS”?

1. Inlining of Iterator Methods

How does Rust Compiler optimize “ITERATORS”?

2. Loop Fusion:

```rust
let result: Vec<_> = (1..100)
    .map(|x| x * 2)
    .filter(|x| x % 3 == 0)
    .collect();
```

The compiler can optimize this into a **single pass** over the range, performing both the **map and filter operations in the same loop**.

How does Rust Compiler optimize “ITERATORS”?

3. Removal of Unused Iterations (Dead Code Elimination)

4. Avoiding Intermediate Allocations

```
let result: Vec<_> = (1..10)
    .map(|x| x + 1)
    .collect();
```

Instead of creating temporary lists, Rust collects elements directly into result, reducing allocation overhead.

Iterator and IntoIterator Traits:

```rust
trait Iterator {
    type Item;
    fn next(&mut self) -> Option<Self::Item> {
        // Login to return an item or None
    }
}
```

“Item” : Type of the value the iterator produces.

”next” : Either returns Some(v) or None

Writing own iterator:

```rust
struct Counter {
    current: i32,
    limit: i32
}

impl Counter {
    fn new(a: i32) -> Self {
        Counter {current:0, limit: a}
    }
}

impl Iterator for Counter {
    type Item = i32;

    fn next(&mut self) -> Option<Self::Item>{
        if self.current < self.limit {
            self.current += 1;
            return Some(self.current-1);
        } else {
            return  None;
        }
    }
}
```

Implement Iterator for Counter:

1. Specifies the type of items the iterator will yield.
2. Required method to implement.

   Each call to next should:

   - Return Some(value) while there are more items to iterate.
   - Return None once the iterator has finished (when current >= limit).

Documenting:

All mark down features are supported.

More Options: https://doc.rust-lang.org/cargo/commands/cargo-doc.html

Thank you !!!