Principles of Programming

20CYS312 - Principles of Programming Languages

S. Shyam Balaji **CH.EN.U4CYS22045** **DATE:** 21/03/25

Lab Exercise Submission

GitHub - shyam150801/principle-of-programming

Contribute to shyam150801/principle-of-programming development by creating an account on GitHub.

https://github.com/shyam150801/principle-of-programming

Objective

The objective of this lab exercise is to learn how to **implement a custom iterator** in Rust by defining a struct (`EvenNumbers`), implementing the **Iterator** trait, and using the **next()** method to generate even numbers. You will also practice using this iterator to print the first **10 even numbers** in the `main()` function.

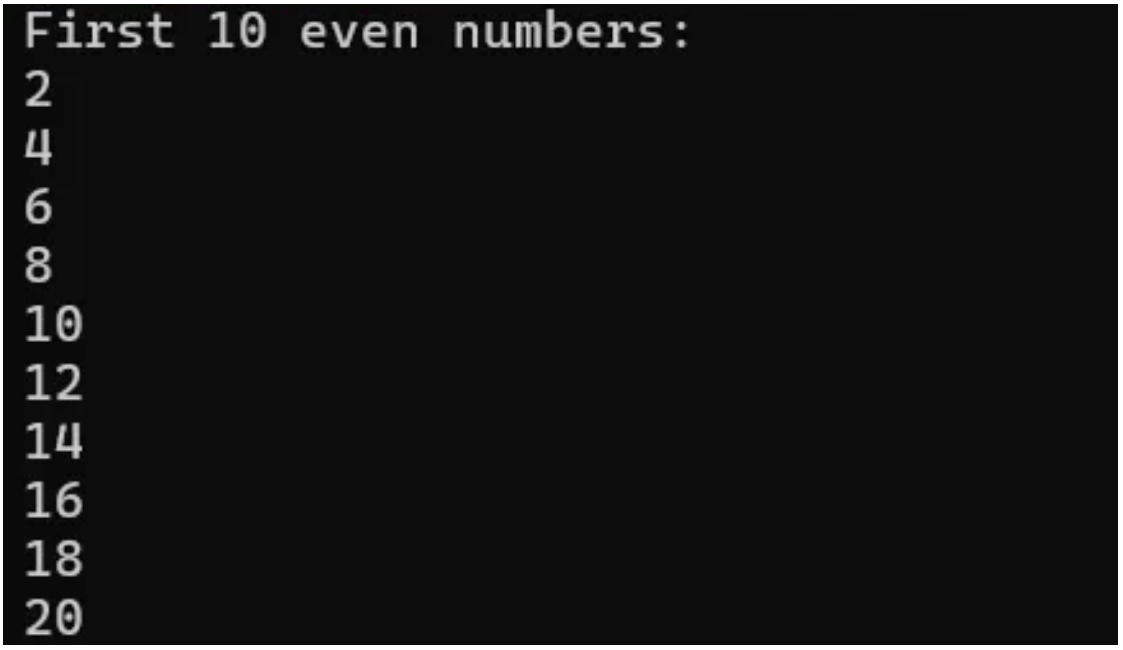
Code

Define a Custom Iterator for Even Numbers

```
// Define a custom iterator for even numbers
struct EvenNumbers {
    current: u32,
    limit: u32,
}

// Implement the Iterator trait for EvenNumbers
impl Iterator for EvenNumbers {
    type Item = u32;
```

```
    fn next(&mut self) -> Option<Self::Item> {
        if self.current > self.limit {
            return None; // Stop when the limit is reached
        }

        let next_even = self.current;
        self.current += 2; // Move to the next even number

        Some(next_even)
    }
}

// Function to create a new EvenNumbers iterator
fn even_numbers(limit: u32) -> EvenNumbers {
    EvenNumbers { current: 2, limit }
}

fn main() {
    println!("First 10 even numbers:");

    // Create an EvenNumbers iterator up to 20
    let even_iter = even_numbers(20);

    // Print the first 10 even numbers
    for num in even_iter.take(10) {
        println!("{}", num);
    }
}
```

Output




```
First 10 even numbers:
2
4
6
8
10
12
14
16
18
20
```

Explanation

1. **Define the EvenNumbers Struct:**
   - This struct holds two fields:
     - `current`: Tracks the current even number.
     - `limit`: Specifies the maximum value the iterator will produce.
2. **Implement the Iterator Trait:**
   - `type Item = u32;` defines the type of values the iterator yields.

- `next()` method:
  - If `current` exceeds `limit`, return `None` (stopping condition).
  - Otherwise:
    - Store the current value.
    - Increment by `2` to get the next even number.
    - Return `Some(next_even)`.

3. **Create the `even_numbers()` Function:**
   - This function initializes and returns an `EvenNumbers` iterator.
4. **Demonstrate the Iterator in `main()`:**
   - Create an iterator that generates even numbers up to `20`.
   - Use `.take(10)` to limit the output to the **first 10 even numbers**.
   - Print each value using a `for` loop.

Conclusion

This program successfully implements a **custom iterator** in Rust. You learned how to:

1. Define a **struct** to track iterator state.
2. Implement the **Iterator** trait and its `next()` method.
3. Use the `.take()` method to **limit** the output from the iterator.
4. Generate and print the **first 10 even numbers** sequentially.

This knowledge is crucial for designing efficient and reusable iterators in Rust.