Name: Sushant Yadav
Reg No: Ch.en.u4cys22067
Course Code: 20cys312
Date: 2025/03/22

Lab Exercise 11: Custom Iterator Implementation

Create a custom iterator named `EvenNumbers` that generates even numbers starting from 2 up to a given limit.

- Implement the `Iterator` trait for the struct.
- Use the `next()` method to return even numbers sequentially.
- Demonstrate the iterator in `main()` by printing the first 10 even numbers.

Objective

The objective of this lab is to implement a custom iterator in Rust that generates even numbers starting from 2 up to a specified limit. By completing this exercise, you will:

- Understand how to create and implement the `Iterator` trait in Rust.
- Learn to use the `next()` method to generate sequence-based outputs.
- Gain hands-on experience with iterators and how they work in Rust.

Code:

```rust
// Define a struct EvenNumbers that will hold the current value and the limit
struct EvenNumbers {
    current: u32,
    limit: u32,
}

// Implement the EvenNumbers struct
impl EvenNumbers {
    // Constructor to create a new EvenNumbers iterator
    fn new(limit: u32) -> Self {
        EvenNumbers {
            current: 2, // Start from 2
            limit,
        }
    }
}

// Implement the Iterator trait for EvenNumbers
impl Iterator for EvenNumbers {
    type Item = u32; // The type of item the iterator will yield (even numbers)

    // Implement the next() method to return the next even number
    fn next(&mut self) -> Option<Self::Item> {
        if self.current <= self.limit {
            let result = self.current;
            self.current += 2; // Move to the next even number
            Some(result) // Return the next even number
        } else {
            None // Stop when the current value exceeds the limit
        }
    }
}

fn main() {
    // Create an EvenNumbers iterator with a limit of 20
    let even_numbers = EvenNumbers::new(20);

    // Use the iterator to print the first 10 even numbers
    for even_number in even_numbers {
        println!("{}", even_number);
    }
}
```

Explanation

1. Understanding Iterators in Rust

In Rust, an iterator is a construct that allows you to process a sequence of elements. The `Iterator` trait provides the `next()` method, which returns the next element in the sequence.

2. Struct Definition

We define a struct `EvenNumbers` that will keep track of the current number and the limit up to which we want to generate even numbers.

3. Implementing the Iterator Trait

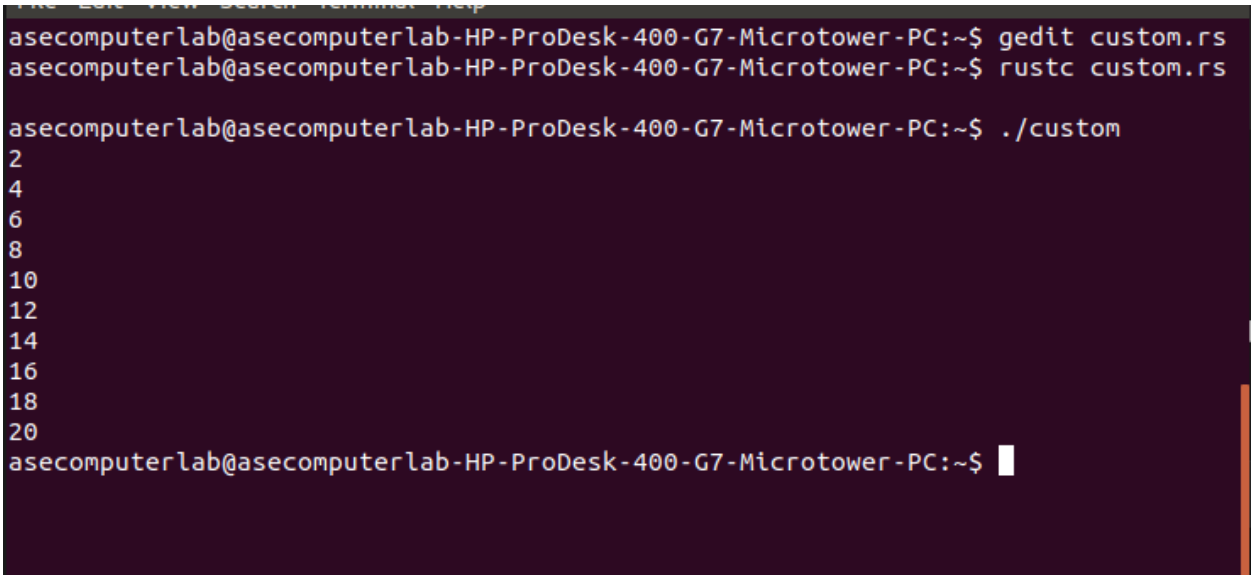
The `Iterator` trait requires us to define the `next()` method. This method will:

- Start from 2.
- Return even numbers sequentially.
- Stop when the specified limit is reached.

4. Demonstrating the Iterator in `main()`

We create an instance of `EvenNumbers` and use a loop to print the first 10 even numbers.

Output:



Conclusion

This exercise demonstrated how to create a custom iterator in Rust by implementing the `Iterator` trait. We learned:

- How to define and manage state within a struct.
- How to override `next()` to generate even numbers.
- How to use iterators in a loop effectively.