WEB3CLUBS FOUNDATION LIMITED

Course Instructor: DR. Cyprian Omukhwaya Sakwa
PHONE: +254723584205 Email: cypriansakwa@gmail.com

Foundational Mathematics for Web3 Builders

Implemented in RUST

Lecture 37

July 25, 2024

The following Rust code shows how to compute and publish a multiplication table for a set of integers that are relatively prime to a given modulus n when multiplied modulo n.

```rust
fn gcd(a: u64, b: u64) -> u64 {
        let mut x = a;
        let mut y = b;
        while y != 0 {
                let temp = y;
                y = x % y;
                x = temp;
        }
        x
}

fn relatively_prime_elements(n: u64) -> Vec<u64> {
        let mut result = Vec::new();
        for i in 1..n {
                if gcd(i, n) == 1 {
                        result.push(i);
                }
        }
        result
}
```

0, 1

(1)

(1,3)

let temp = 3
y = 1
x = 3

1 let temp = 1
y = 0
x = 1

gcd = x = 1

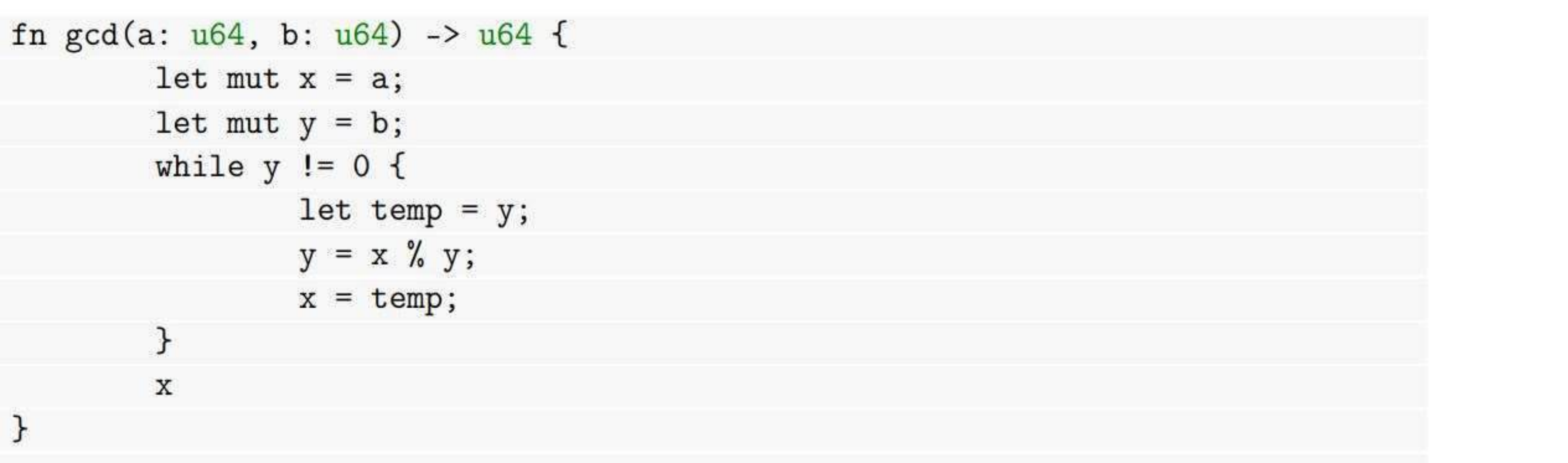
```
fn multiplication_table(group: &[u64], mod_value: u64) {
    println!("Multiplication␣Table␣(mod␣{}):", mod_value);
    // Print the header row
    print!("␣␣␣␣");
    for &b in group {
        print!("{:4}", b);
    }
    println!();

    // Print the table
    for &a in group {
        print!("{:2}␣", a);
        for &b in group {
            print!("{:4}", (a * b) % mod_value);
        }
        println!();
    }
}

fn main() {
    let n = 16; // Example value
    let group = relatively_prime_elements(n);
    multiplication_table(&group, n);
}
```

5

	1	2	3	4
1.	1	2	3	4
2.	2	4	1	3
3.				
4				

Understanding the Rust code

1. Function: gcd. Computes the greatest common divisor $\gcd$ of two numbers using the Euclidean algorithm.



```
fn gcd(a: u64, b: u64) -> u64 {
        let mut x = a;
        let mut y = b;
        while y != 0 {
                let temp = y;
                y = x % y;
                x = temp;
        }
        x
}
```

- Initialize x with a and y with b.
- Enter a while loop that continues as long as y is not zero.
- Inside the loop

  ✓ Store the value of y in a temporary variable temp.

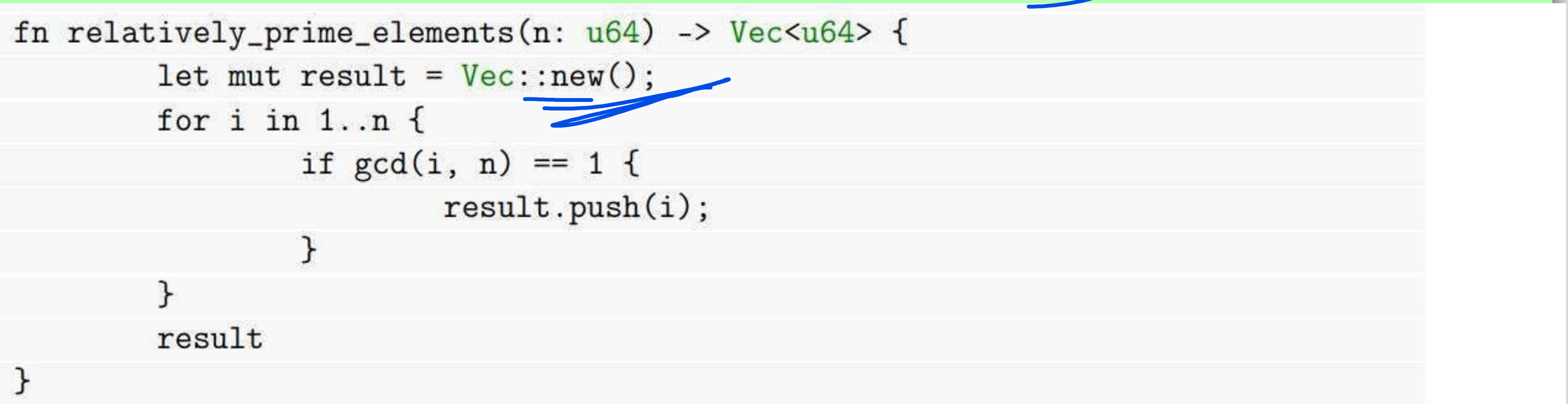
  ✓ Update y to $x\%y$ (the remainder when x is divided by y)

Understanding the Rust code (conti...)

✓ Update x to temp (the old value of y).

- When y becomes zero, x contains the gcd of a and b.

(2) Function: relatively_prime_elements. Finds all integers from 1 to n to $n-1$ that are relatively prime to n.



```
fn relatively_prime_elements(n: u64) -> Vec<u64> {
        let mut result = Vec::new();
        for i in 1..n {
                if gcd(i, n) == 1 {
                        result.push(i);
                }
        }
        result
}
```

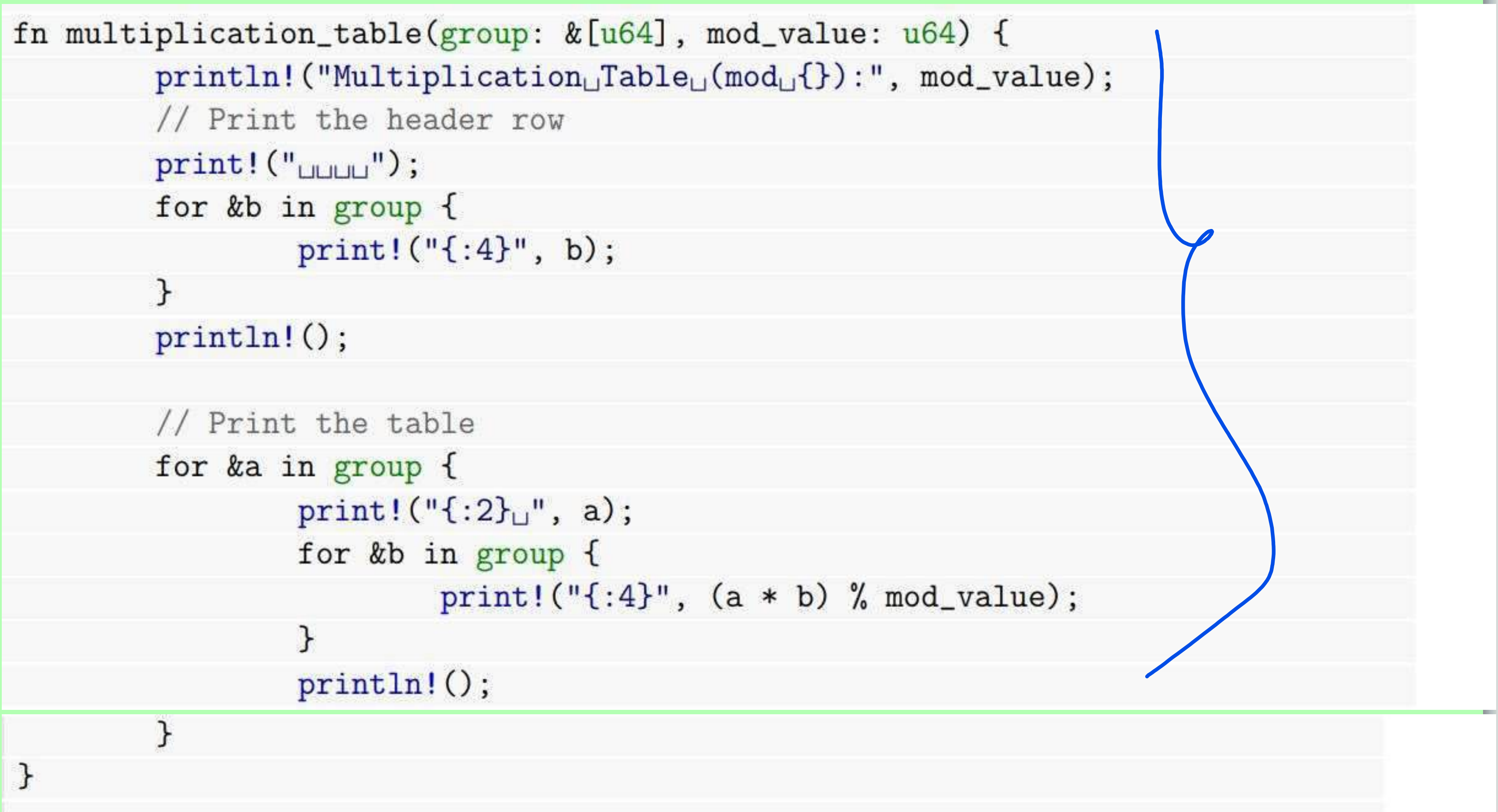
- Create an empty vector result to store the relatively prime elements.

Understanding the Rust code (conti...)

- Loop through each integer i from 1 to n to $n-1$:
  - ✓ Check if i is relatively prime to n by computing $\gcd(i, n)$ and checking if it equals 1.
  - ✓ If true, add i to the result vector.
  - ✓ Return the result vector, which contains integers that are relatively prime to n.

Understanding the Rust code (conti...)

(3) Function: multiplication_table. Creates and prints a multiplication table for a specified set of integers modulo mod_value.

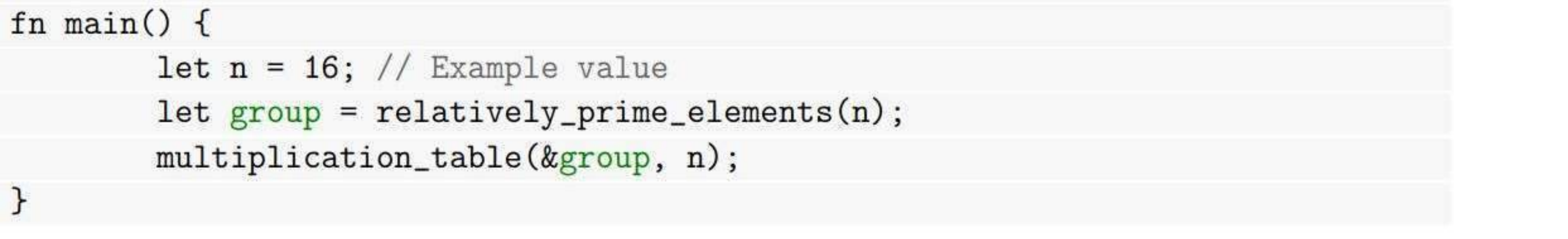


```
fn multiplication_table(group: &[u64], mod_value: u64) {
        println!("Multiplication␣Table␣(mod␣{}):", mod_value);
        // Print the header row
        print!("␣␣␣␣");
        for &b in group {
                print!("{:4}", b);
        }
        println!();

        // Print the table
        for &a in group {
                print!("{:2}␣", a);
                for &b in group {
                        print!("{:4}", (a * b) % mod_value);
                }
                println!();
        }
}
```

Understanding the Rust code (conti...)

- Print the heading for the multiplication table.
- Print the header row with the elements of the group formatted to be right-aligned in a field of width 4.
- For each element a in the group, print a row where each entry is the product (a * b) % mod_value for each element b in the group.
- Each row is formatted so that each entry is right-aligned in a field of width 4.

(4) Main Function. Demonstrates the use of the above functions with a specific.



```
fn main() {
        let n = 16; // Example value
        let group = relatively_prime_elements(n);
        multiplication_table(&group, n);
}
```

Understanding the Rust code (conti...)

- sets the modulus value.
- Call relatively_prime_elements(n) to get all integers from 1 to $n-1$ that are relatively prime to n.
- Call multiplication_table(&group, n) to print the multiplication table for these integers under multiplication modulo n.