

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 .

```
1 fn gcd(a: u64, b: u64) -> u64 {
2     let mut x = a;
3     let mut y = b;
4     while y != 0 {
5         let temp = y;
6         y = x % y;
7         x = temp;
8     }
9     x
10 }
11
12 fn relatively_prime_elements(n: u64) -> Vec<u64> {
13     let mut result = Vec::new();
14     for i in 1..n {
15         if gcd(i, n) == 1 {
16             result.push(i);
17         }
18     }
19     result
20 }
21
```

0, 1

(1)

(1, 3)

b) temp = 3
y = 1
x = 3

1. let temp = 1
y = 0
x = 1

$\gcd(x, n) = 1$

```

22 fn multiplication_table(group: &[u64], mod_value: u64) {
23     println!("Multiplication Table(mod {}):", mod_value);
24     // Print the header row
25     print!("{}", "    ");
26     for &b in group {
27         print!("{:4}", b);
28     }
29     println();
30
31     // Print the table
32     for &a in group {
33         print!("{:2} ", a);
34         for &b in group {
35             print!("{:4}", (a * b) % mod_value);
36         }
37         println();
38     }
39 }
40
41 fn main() {
42     let n = 16; // Example value
43     let group = relatively_prime_elements(n);
44     multiplication_table(&group, n);
45 }

```

Handwritten annotations and a multiplication table for mod 16.

Header row (line 25):

	1	2	3	4
--	---	---	---	---

Table body (lines 32-37):

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

Handwritten notes: A large blue bracket on the right side of the code block. A blue circle around the expression `(a * b) % mod_value` in line 35. A blue checkmark next to line 43.

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 $\text{gcd}(i, n)$ 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 $(a * b) \% \text{mod_value}$ 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 .