

Лабораторная работа №5

**Дисциплина: Математические основы защиты информации и
информационной безопасности**

Аветисян Давид Артурович

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	11

List of Tables

List of Figures

3.1	Тест Ферма на языке Julia	7
3.2	Тест Ферма на языке Julia	7
3.3	Поиск символа Якоби на языке Julia	8
3.4	Тест Соловья-Штрассена на языке Julia	8
3.5	Тест Соловья-Штрассена на языке Julia	9
3.6	Тест Миллера-Рабина на языке Julia	9

1 Цель работы

Познакомиться с вероятностными алгоритмами проверки чисел на простоту.

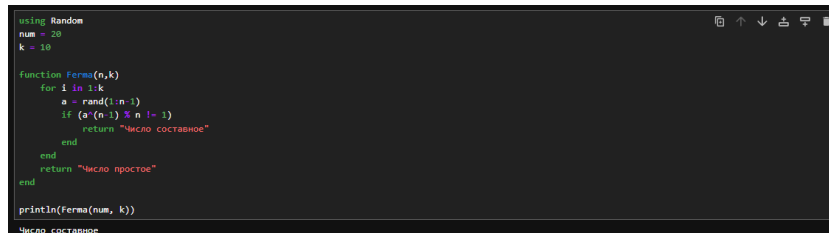
2 Задание

1. Реализовать тест Ферма.
2. Реализовать алгоритм вычисления символа Якоби.
3. Реализовать тест Соловья-Штрассена.
4. Реализовать тест Миллера-Рабина.

3 Выполнение лабораторной работы

Данная работа была выполнена на языке Julia.

1) Для реализации теста Ферма была написана следующая программа.



```
using Random
num = 29
k = 10

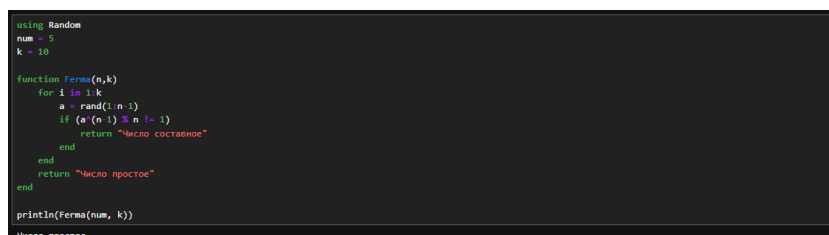
function Fermat(n,k)
    for i in 1:k
        a = rand(1:n-1)
        if (a^(n-1) % n != 1)
            return "Число составное"
        end
    end
    return "Число простое"
end

println(Fermat(num, k))
Число простое
```

Figure 3.1: Тест Ферма на языке Julia

В данной программе: - 2-3 строки: задание числа, которое нужно проверить на простоту, и количество проверок. - 5 строка: задание функции. - 6 строка: цикл проверки выполняется k раз. - 7 строка: берётся случайно число a в диапазоне $[1, n - 1]$. - 8 строка: проводим проверку условия, при невыполнении сразу завершаем работу. - 9-13 строки: выводим результат, закрываем функцию.

Мы можем видеть результат для двух случаев на рисунках выше и ниже. Программа работает верно.



```
using Random
num = 5
k = 10

function Fermat(n,k)
    for i in 1:k
        a = rand(1:n-1)
        if (a^(n-1) % n != 1)
            return "Число составное"
        end
    end
    return "Число простое"
end

println(Fermat(num, k))
Число простое
```

Figure 3.2: Тест Ферма на языке Julia

2) Для реализации поиска символа Якоби была написана следующая программа.

```
[19]: function Jacobi(a,n)
    if ! (n > a > 0 && n % 2 == 1)
        return 0
    end
    s = 1
    while a != 0
        while a % 2 == 0
            a /= 2
        end
        k = n % 8
        if k == 3 || k == 5
            s = -s
        end
        a, n = n, a
        if a % 4 == 3 && n % 4 == 3
            s = -s
        end
        a %= n
    end
    if n == 1
        return s
    else
        return 0
    end
end

println("Символ Якоби ", Jacobi(7,33))

Символ Якоби -1
```

Figure 3.3: Поиск символа Якоби на языке Julia

В данной программе: - 1-5 строки: задание функции, проверка условия для вычисления символа Якоби. - 6-25 строки - реализация алгоритма: проверка трёх условия и действия согласно этим условиям: смена знака символа при четном и нечетном k , проверка остатков от деления. - 26 строка: вывод результата работы программы. В данном случае я вычислял Якоби 7 и 33. Вывод представлен на рисунке выше.

3) Для реализации теста Соловья-Штрассена была написана следующая программа.

```
[26]: using Random

function S_Sh(n,k)
    for i in 1:k
        a = rand{Int64}(n-1)
        r = a * ((n-1) % 2) % n
        if r != 1 && r != n-1
            return "Число составное"
        end
    end
    s = Jacobi(n,a)
    if r == s % n
        return "Число составное"
    end
    return "Число простое"
end

println(S_Sh(num, k))

Число составное
```

Figure 3.4: Тест Соловья-Штрассена на языке Julia

В данной программе: - 3 строка: задаём функцию. - 4 строка: повторим проверку k раз - 5-16 строки - реализация алгоритма: выбираем случайное число

a , вычисляем число r по формуле в строке 6, а затем проверяем получившееся значение на два условия. Если оно не проходит, проверку, то сразу заканчиваем работу программы. Далее следует ещё одна проверка условия в строке 11, при провале также завершаем работу. - 18 строка: вывод на экран. Результат работы программы с числами 20 и 5 и 10-ю проверками.

Мы можем видеть результат для двух случаев на рисунках выше и ниже. Программа работает верно.

```
[29]: using Random

function S_Sh(n,k)
    for i in 1:k
        a = rand(2:(n-1))
        r = a^((n-1) ÷ 2) % n
        if r != 1 && r != n - 1
            return "Число составное"
        end
        s = Jacobi(n,a)
        if r == s % a
            return "Число составное"
        end
    end
    return "Число простое"
end

println(S_Sh(num, k))

Число простое
```

Figure 3.5: Тест Соловья-Штрассена на языке Julia

4) Для реализации теста Миллера-Рабина была написана следующая программа.

```
function M_R(n, k)
    if n == 2
        return "Число простое"
    end
    if n % 2 == 0
        return "Число составное"
    end
    r, s = 0, n - 1
    while s % 2 == 0
        r += 1
        s ÷= 2
    end
    for _ in 1:k
        a = rand(2:(n-1))
        x = powermod(a, s, n)
        if x == 1 || x == n - 1
            continue
        end
        for i in 1:(r-1)
            x = powermod(x, 2, n)
            if x == n - 1
                break
            else
                return "Число составное"
            end
        end
    end
    return "Число простое"
end

println(M_R(num, k))

Число составное
```

Figure 3.6: Тест Миллера-Рабина на языке Julia

Данная программа работает рекурсивно, рассматривая 4 случая: - 3 строка: задаём функцию. - 4-9 строки: отсеивание числа 2 и остальных четных чисел. - 11-29 строки - реализация алгоритма: выбираем случайное число a и вычисляем

число x по формуле в строке 17. При условии в строке 18 выполняем дополнительные действия - вычисление остатка от деления квадрата x на проверяемое число. Если число прошло все проверки k раз, мы определяем его как “вероятно, простое”.

Результат работы программы с числом 20 и 10-ю проверками представлен на рисунке выше.

4 Выводы

Я познакомился с вероятностными алгоритмами проверки чисел на простоту.