# Отчет по лабораторной работе 5

**По предмету мат. основы защиты информации**

Студент: Дидусь Кирилл Валерьевич, 1132223499
Группа: НПМмд-02-22
Преподаватель: Кулябов Дмитрий Сергеевич,
д-р.ф.-м.н., проф.

Москва, 2022

# Содержание

# Список иллюстраций

# Список таблиц

# 1 Цель работы

Целью данной лабораторной работы является ознакомление с алгоритмом для вероятностной проверки числа на простоту.

# 2 Задание

Реализовать алгоритмом для вероятностной проверки числа на простоту в программном виде.

# 3 Выполнение лабораторной работы

В ходе выполнения лабораторной работы было реализовано 3 вариации алгоритма для вероятностной проверки числа на простоту, использующих различные тесты.

Среди них:

- тест Ферма
- тест Соловэя-Штрассена
- тест Миллера-Рабина

Программный код представлен в качестве листинга в конце отчета.

# 4 Выводы

Таким образом, была достигнута цель, поставленная в начале лабораторной работы: я ознакомился с алгоритмом для вероятностной проверки числа на простоту, а так же мне удалось реализовать варации этого алгоритма на языке программирования Python.

# 5 Листинг программы

```
# Проверка чисел на простоту является составной частью
# алгоритмов генерации простых чисел, применяемых в криптографии с открытым ключо
# АПП бывают вероятностные и детерминированные. Рассмотрим вероятностные алгоритм

import random as rnd

def dividers_count(n,divider):
    i = 0
    n_copy = n
    while (n_copy % divider == 0):
        i += 1
        n_copy /= divider
    return i

def jacobi(n,a):
    if n<3:
        print("n can't be lower than 5")
        return
    g = 1
    res, s = 0,0
    n_copy = n
    a_copy = a
```

```
    while(True):
        if(a_copy == 0):
            res = 0
            return res
        elif(a_copy == 1):
            res = g
            return res
        k = dividers_count(a_copy,2)
        a1 = a_copy / pow(2,k)
        if(k%2 ==0):
            s = 1
        else:
            if ((n_copy % 8 == 1) | (n_copy % 8 == -1)):
                s = 1
            elif ((n_copy % 8 == 3) | (n_copy % 8 == -3)):
                s = -1
        if(a1==1):
            res = g*s
            return res
        if((n_copy % 4 == 3) & (a1%4==3)):
            s = -s
        a_copy = n_copy % a1
        n_copy = a1
        g = g*s

def ferma_test(n): #n>=5
    if n<5:
        print("n can't be lower than 5")
        return
```

```
    a = rnd.randrange(2,n-2)
    r = pow(a,(n-1))%n
    if(r==1):
        return True
    else:
        return False

def solovayStrassen_test(n):
    if n<5:
        print("n can't be lower than 5")
        return
    a = rnd.randrange(2,n-2)
    r = pow(a,((n-1)/2)%n)
    s = jacobi(n,a)
    if (r % n == s):
        return False
    else:
        return True

def millerRabin_test(n):
    if n<5:
        print("n can't be lower than 5")
        return
    s = dividers_count(n-1,2)
    r = (n-1) / pow(2,s)
    a = rnd.randrange(2,n-2)
    y = pow(a,r)%n
    if((y != 1) & (y != n-1)):
        j = 1
```

```
        if ((j < s - 1) & (y != n - 1)):
            y = y*y % n
            if (y == 1):
                return False
            j += 1
        if (y != n-1):
            return False
    return True


def is_prime(n):
    for i in range(20):
        # если тест возрващают True 20 раз, n очень вероятно простое число.
        if((ferma_test(n)) & True):
            continue
        else:
            return False
    return True

# if true - вероятно простое
# if false - составное
n = int(input("введите n: "))
print(is_prime(n))
```