**Лабораторная работа № 3**

**ОСНОВЫ ТЕОРИИ ЧИСЕЛ И ИХ ИСПОЛЬЗОВАНИЕ В КРИПТОГРАФИИ**

**Цель**: приобретение практических навыков выполнения операций с числами для решения задач в области криптографии и разработка приложений для автоматизации этих операций.

**Задачи:**

1. Закрепить теоретические знания по высшей арифметике.

2. Научиться практически решать задачи с использованием простых и взаимно простых чисел, вычислений по правилам модулярной арифметики и нахождению обратных чисел по модулю.

3. Ознакомиться с особенностями реализации готового программного средства L\_PROST и особенностями выполнения с его помощью операций над простыми числами.

4. Разработать приложение для реализации указанных преподавателем операций с числами.

5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения эксперимента с использованием приложения и результатов эксперимента.

**Практическое задание.**

Рекомендация! Перед выполнением практического задания можно познакомиться с функционалом программного средства L\_PROST, являющегося приложением на компакт-диске к пособию [6], особенностями выполнения с его помощью операций над простыми числами.

На рис.1.1 представлено основное диалоговое окно программы после запуска исполнительного файла L\_PROST.EXE. Как видно, средство позволяет генерировать простые числа, осуществлять проверку чисел на простоту и определять простые числа в заданном интервале.

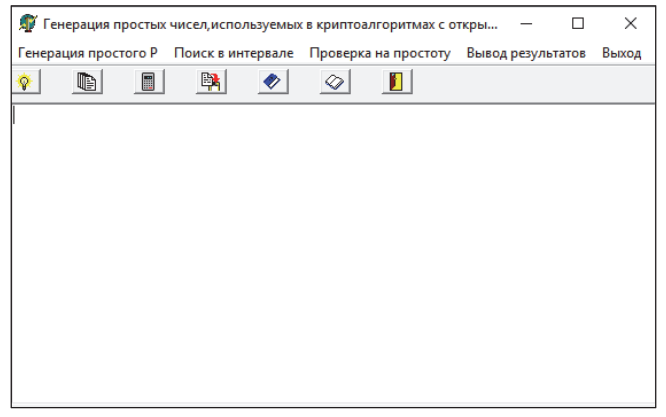


Рисунок 1.1. Основное диалоговое окно программного средства L\_PROST

1. Используя L\_PROST, найти все простые числа в интервале [2, n]. Значение n соответствует варианту из табл. 1.2, указанному преподавателем.

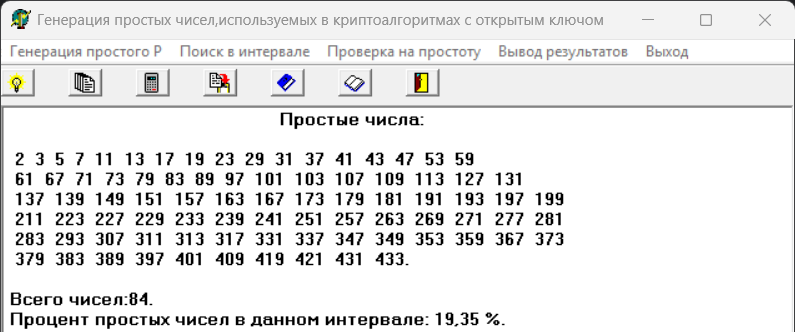


Рисунок 1.2 простые числа в интервале [2, n], где n 433

Подсчитать количество простых чисел в указанном интервале. Сравнить это число с n/ln(n) (см. выше пример 15).

2. Повторить п. 1 для интервала [m, n].

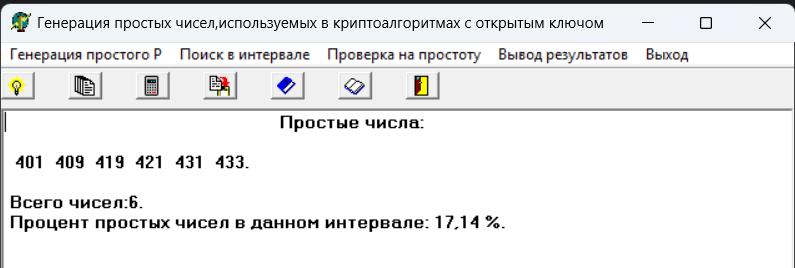


Рисунок 1.3 простые числа в интервале [m, n], где m 399, а n 433

Сравнить полученные результаты с «ручными» вычислениями, используя «решето Эратосфена» (см. примеры 11 и 12). 3. Записать числа m и n в виде произведения простых множителей (форма записи – каноническая).

**Пример 11**

Запишем числа от 399 до 433: 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433

S = 2, тогда удалим числа кратные 2

Останутся числа: 399, 401, 403, 405, 407, 409, 411, 413, 415, 417, 419, 421, 423, 425, 427, 429, 431

Далее S = 3, удалим числа, которые кратны 3

Останутся числа: 401, 409, 419, 421, 431, 433

Далее S = 5, удалим числа, которые кратны 5

Таких чисел нет, останутся прежние числа

Далее S = 7, удалим числа, которые кратны 7

Таких чисел нет, останутся прежние числа

Далее S = 11, удалим числа, которые кратны 11

Таких чисел нет, останутся прежние числа

Далее S = 13, удалим числа, которые кратны 13

Таких чисел нет, останутся прежние числа

Далее S = 17, удалим числа, которые кратны 17

Таких чисел нет, останутся прежние числа

Итоговый набор: 401, 409, 419, 421, 431, 433

Число 399, делится на 3, далее число 133 делится на 7 и получается 19, а число 19 простое, m = 399 = 3\*7\*19, число n = 433 является простым

4. Проверить, является ли число, состоящее из конкатенации цифр m ǀǀ n (табл. 1.2), простым.

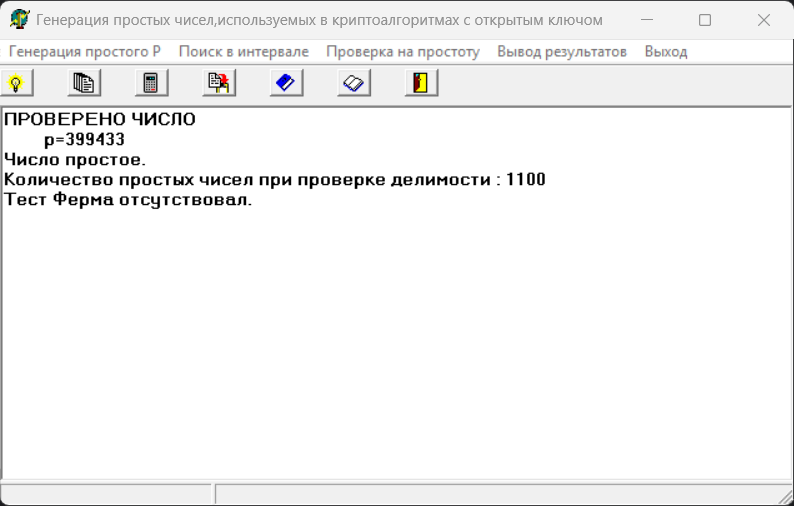


Рисунок 1.4 проверка числа 399433 на простоту

5. Найти НОД (m, n).

Найдём НОД числа m = 399, n = 433

Так как число 433 является простым и не делится без остатка на 399, то наибольшим общим делителем для таких чисел будет число 1.

**Основное задание**

6. Разработать авторское приложение в соответствии с целью лабораторной работы. Приложение должно реализовывать следующие операции:

• вычислять НОД двух либо трех чисел;

• выполнять поиск простых чисел.

Функция считающая НОД двух чисел представлена в листинге 1.1.

function greatestCommonDivisorTwoNums(num1:number, num2:number):number{

let remainder:number

while (true) {

if (num1 > num2) {

[num1, num2] = [num2, num1];}

remainder = num2 % num1;

if (remainder === 0) {

return num1;}

num2 = remainder;

}

}

Листинг 1.1 функция greatestCommonDivisorTwoNums

Функция считающая НОД трёх чисел представлена в листинге 1.2.

function greatestCommonDivisor(num1:number, num2:number, num3?:number):number{

if(num3){

return greatestCommonDivisorTwoNums(greatestCommonDivisorTwoNums(num1, num2), greatestCommonDivisorTwoNums(num2, num3))

}

else{

return greatestCommonDivisorTwoNums(num1, num2)}}

Листинг 1.2 функция greatestCommonDivisor

Функция, определяющая простоту числа представлена в листинге 1.3.

function IsPrime(num: number): boolean {

if (num <= 1) return false;

for (let i = 2; i <= Math.sqrt(num); i++) {

if (num % i === 0) {

return false;

}

}

return true;

}

Листинг 1.3 функция IsPrime

7. С помощью созданного приложения выполнить задания по условиям п. 1 и 2.

Код, демонстрирующий разработанные функции представлен в листинге 1.4.

console.log(greatestCommonDivisor(num1,num2))

console.log(greatestCommonDivisor(num1,num2, num3))

console.log(IsPrime(10))

console.log(IsPrime(7))

console.log(IsPrime(777))

console.log(IsPrime(101))

let primeNumbers:number[] = []

primeNumbers = primeRange(2,433)

let n:number = primeNumbers.length

console.log(primeNumbers)

console.log(n, n/Math.log(n))

primeNumbers = primeRange(399,443)

n = primeNumbers.length

console.log(primeNumbers)

console.log(n, n/Math.log(n))

Листинг 1.4 Демонстрационный код

Результат работы кода представлен на рисунке 1.5.

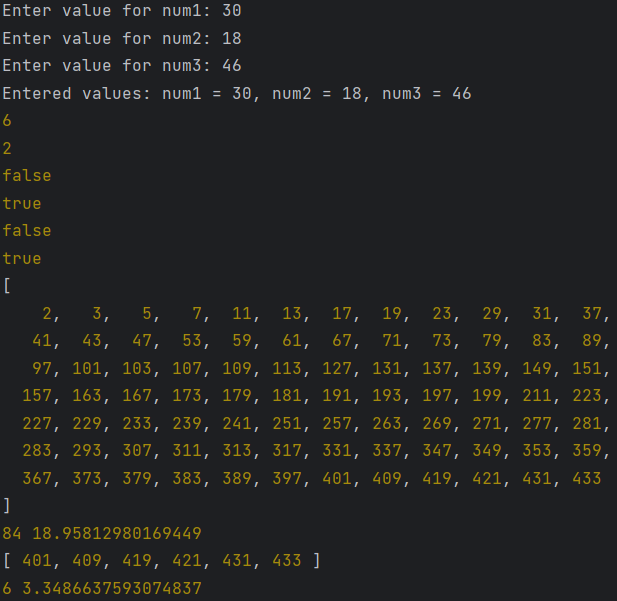


Рисунок 1.5 Результат работы кода

8. Результаты выполнения работы оформить в виде отчета по установленным правилам.

**Варианты задания**

