Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40 01 01 Программное обеспечение информационных технологий

**Отчёт по лабораторной работе №3**

**«Основы теории чисел и их использование в криптографии»**

Исполнитель:

Студент 3 курса группы 4

Гурина К. С.

Руководитель:

Ассистент Сазонова Д. В.

1. **Цель и задачи работы**

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.

**2. Теоретические сведения**

В основе современной криптографии лежит теория чисел. Теория чисел, или высшая арифметика, – раздел математики, изучающий натуральные числа и иные похожие величины.

Множество всех целых чисел (обозначим буквой Z) есть набор всех действительных чисел без дробной части: {..., –3, –2, –1, 0, 1, 2, 3, ...}.

Натуральные числа являются подмножеством целых чисел и образуют множество N: {1, 2, 3, ...}.

Делимость – одно из основных понятий теории чисел. Если для некоторого целого числа a и натурального числа b существует целое число q, при котором bq = a, то говорят, что число a делится на b. В этом случае b называется делителем числа a, а a называется кратным числу b.

Делитель a называется собственным делителем числа b, если 1 < |a| < |b|, и несобственным – в противном случае.

Всякое целое число а можно представить с помощью положительного целого числа b равенством вида а = bq + r, 0 ≤ r ≤ b. Число q называется неполным частным, а число r – остатком от деления а на b.

Каждое натуральное число, большее единицы, делится по крайней мере на два числа: на 1 и на само себя.

Если число не имеет делителей, кроме самого себя и единицы, то оно называется простым, а если у числа есть еще делители, то составным.

Натуральное число n называется простым, если n > 1 и не имеет положительных делителей, отличных от 1 и n.

Перечислим несколько важных свойств простых чисел.

Свойство 1. Любое составное число представляется уникальным образом в виде произведения простых чисел; иначе еще говорят, что разложение числа на простые множители однозначно.

Свойство 2. Простых чисел бесконечно много, причем существует примерно n/ln(n) простых чисел, меньших числа n.

Свойство 3. Наименьший простой делитель составного числа n не превышает √n, поэтому для проверки простоты числа достаточно проверить его делимость на 2 и на все нечетные (а еще лучше простые) числа, не превосходящие √n; как видим, данное свойство коррелирует со свойством 1 собственного делителя.

Свойство 4. Любое четное число, большее 2, представимо в виде суммы двух простых чисел, а любое нечетное, большее чем 5, представимо в виде суммы трех простых чисел.

Свойство 5. Для любого натурального n, большего 1, существует хотя бы одно простое число на интервале от n до 2n.

Натуральное число n называется составным, если n > 1 и имеет, по крайней мере, один положительный делитель, отличный от 1 и n.

Единица не считается ни простым числом, ни составным.

**3. Практическая часть**

**Практическое задание:**

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

4. Проверить, является ли число, состоящее из конкатенации цифр m ǀǀ n (табл. 1.2), простым. 5. Найти НОД (m, n).

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.

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

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

**Ход работы**

Для создания собственного приложения, с помощью которого можно вычислять НОД двух либо трех чисел и выполнять поиск простых чисел был разработан ряд функций. Для вычисления НОД с использованием алгоритма Евклида была разработана функция NOD, код которой представлен на рисунке 3.1.

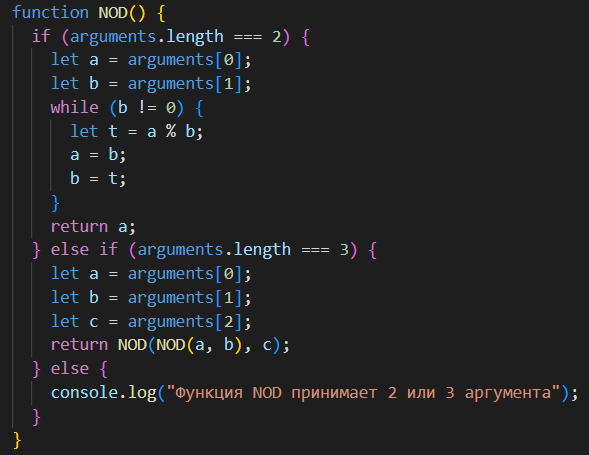


Рисунок 3.1 – Функция для вычисления НОД двух и трёх чисел

Функция может принимать 2 или 3 параметра. При передаче этой функции двух чисел, она вернет НОД этих чисел, а при передаче трёх сначала вычислит НОД первых двух, а потом вернет НОД результата и третьего параметра.

Результат работы функции представлен на рисунке 3.2.

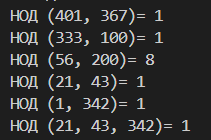


Рисунок 3.2 – Результат работы функции NOD

Ручным способом нахождение НОД(m, n) можно представить следующим образом:

401 = 367 \* 1 + 34

367 = 34 \* 10 + 27

34 = 27 \* 1 + 7

27 = 7 \*3 + 6

7 = 6 \* 1 + 1

6 = 1 \* 6 + 0

Таким образом НОД(401, 367) = 1, это значит, что числа 401 и 367 взаимно простые.

Для поиска всех простых чисел на заданном интервале была разработана функция GetPrimeNumbers. Код функции GetPrimeNumbers можно увидеть на рисунке 3.3.



Рисунок 3.3 – Код функции GetPrimeNumbers

Функции можно передать 1 или 2 параметра. При передаче функции GetPrimeNumbers одного числа n она вернет все простые числа в интервале [2, n], а при передаче двух чисел m и n – в интервале [m, n]. Результат работы функции представлен на рисунке 3.4.

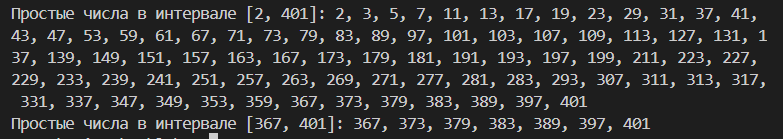


Рисунок 3.4 – Результат работы функции GetPrimeNumbers

Проверим работу функции GetPrimeNumbers ручным способом. Найдем простые числа в интервале [367, 401].

Воспользуемся свойством 3 простых чисел и вычислим √401 ≈ 20,02, т. е. меньше 20. Запишем числа из заданного диапазона и удалим последовательно все числа, делящиеся на простые числа от 2 до 20.

Такими простыми числами являются: 2, 3, 5, 7, 11, 13, 17,19.

Числа в интервале [367, 401]:

367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401

Шаг 1: удалим из списка числа с учетом s = 2:

367 369 371 373 375 377 379 381 383 385 387 389 391 393 395 397 399 401

Шаг 2: удалим из списка числа с учетом s = 3:

367 371 373 377 379 383 385 389 391 395 397 401

Шаг 3: удалим из списка числа с учетом s = 5:

367 371 373 377 379 383 389 391 397 401

Шаг 4: удалим из списка числа с учетом s = 7:

367 373 379 383 389 397 401

После выполнения всех операций в «решете» останутся числа:

367 373 379 383 389 397 401

Чтобы подсчитать количество простых чисел в указанном интервале и сравнить это число с n/ln(n) разработана функция countPrimesInRange, код которой представлен на рисунке 3.5.

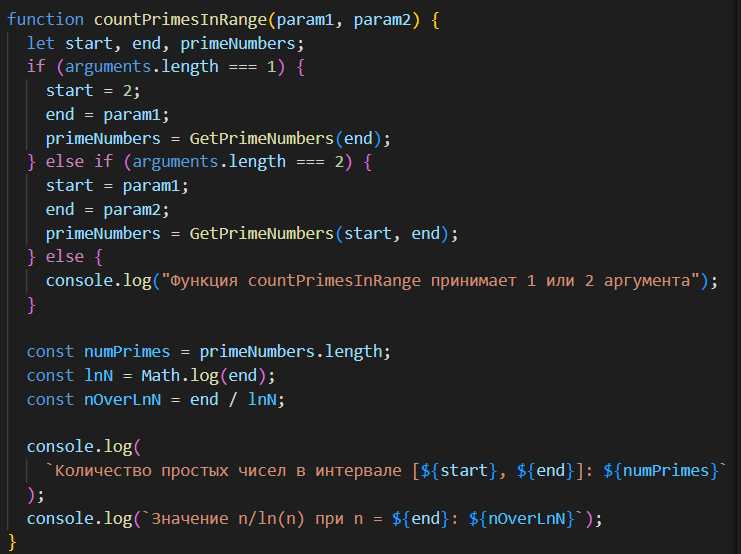


Рисунок 3.5 – Код функции countPrimesInRange

**Вывод**

В данной лабораторной работе мы ознакомились с основами теории чисел и их использованием в криптографии, а также приобрели практические навыки выполнения операций с числами для решения задач в области криптографии и разработали приложение для автоматизации этих операций.