# **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

**Теоретические сведения**

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

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

Определение 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. При этом используются следующие обозначения:

a ⋮ b – a делится на b, или b | a – b делит a.

Из последнего определения следует, что:

• любое натуральное число является делителем нуля;

• единица является делителем любого целого числа;

• любое натуральное число является делителем самого себя.

Определение 4. Делитель a называется собственным делителем числа b, если 1 < |a| < |b|, и несобственным – в противном случае.

Определение 5. 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 отделения а на b.

Определение 6. Натуральное число n называется простым, если n > 1 и не имеет положительных делителей, отличных от 1 и n.

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 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.

Определение 7. Натуральное число n называется составным, если n > 1 и имеет по крайней мере один положительный делитель, отличный от 1 и n.

Определение 8. Если два простых числа отличаются на 2, то их называют числами-близнецами.

Определение 9. Наибольшее целое число, которое делит без остатка числа a и b, называется наибольшим общим делителем этих чисел – НОД (a, b).

Определение 10. Взаимно простыми являются целые числа, наибольший общий делитель которых равен 1.import math  
import numpy as np  
import time  
  
  
def main():  
 print("----- Наибольший общий делитель -----")  
 list\_number = [140, 420, 280, 140, 140]  
 print("Список чисел:", list\_number)  
 gcd = GCD(list\_number)  
 print("НОД чисел:", gcd, "\n")  
  
 print("----- Поиск простых чисел -----")  
 number = 1000  
 listPrimeNumber = SearchForPrimeNumbers(number)  
 print("Простые числа", listPrimeNumber[0])  
 print("Простые числа по формуле n/ln(n):", listPrimeNumber[1])  
 print("Простые числа действительно:", listPrimeNumber[2], "\n")  
  
 print("----- Простое ли число??? -----")  
 primeNumber = 24  
 IsAPrimeNumber(primeNumber)  
  
 print("\n----- Делители числа -----")  
 divisibleNumber = 27  
 DecompositionIntoPrimeNumbers(divisibleNumber)  
  
 print("\n----- Разложение на простые множители -----")  
 primeFactorsNumber = 260  
 DetermineThePrimeFactorsOfANumber(primeFactorsNumber)  
  
  
def GCD(list\_number): # greatest common divisor  
 list\_division = []  
 current\_position = 0  
  
 for i in range(len(list\_number) - 1):  
 list\_division.clear()  
 current\_divisible = min(list\_number[0], list\_number[1])  
 current\_remains = max(list\_number[0], list\_number[1]) % min(list\_number[0], list\_number[1])  
  
 list\_division.append((current\_divisible, current\_remains))  
 while current\_remains != 0:  
 current\_divisible = list\_division[current\_position][1]  
 current\_remains = list\_division[current\_position][0] % list\_division[current\_position][1]  
 list\_division.append((current\_divisible, current\_remains))  
 current\_position += 1  
 current\_position = 0  
  
 list\_number.pop(0)  
 list\_number.pop(0)  
 list\_number.append(list\_division[len(list\_division) - 1][0])  
  
 return list\_division[len(list\_division) - 1][0]  
  
  
def SearchForPrimeNumbers(number = 0):  
 if (number != 0):  
 traversableArray = np.arange(2, number, 1)  
 passingElement = 0  
 while passingElement != len(traversableArray):  
 for passableElement in range(passingElement, len(traversableArray)):  
 if (traversableArray[passableElement] % traversableArray[passingElement] == 0 and traversableArray[passableElement] != traversableArray[passingElement]):  
 traversableArray[passableElement] = 0  
 for i in range(len(traversableArray)):  
 if (traversableArray[i] == 0):  
 for j in range(i + 1, len(traversableArray), 1):  
 traversableArray[i] = traversableArray[j]  
 if (traversableArray[i] != 0):  
 traversableArray[j] = 0  
 break  
 if (any(traversableArray[i + 1:len(traversableArray)]) == 0):  
 traversableArray = traversableArray[:i + 1]  
 break  
 passingElement += 1  
  
 listPrimeNumber = []  
 for i in range(len(traversableArray)):  
 if (traversableArray[i] != 0):  
 listPrimeNumber.append(traversableArray[i])  
  
 return listPrimeNumber, round(number / math.log(number, math.exp(1))), len(listPrimeNumber)  
  
  
def SearchForPrimeNumbersOnTheInterval(listNumber = []):  
 if(len(listNumber) == 2):  
 listPrimeNumber = []  
 traversableArray = np.arange(listNumber[0], listNumber[1] + 1, 1)  
  
 primeNumbersUpToUpperBound = SearchForPrimeNumbers(math.ceil(math.pow(listNumber[1], 0.5)))[0]  
  
 for i in range(len(traversableArray)):  
 for j in range(len(primeNumbersUpToUpperBound)):  
 if(traversableArray[i] % primeNumbersUpToUpperBound[j] == 0 and traversableArray[i] != primeNumbersUpToUpperBound[j]):  
 break  
 else:  
 listPrimeNumber.append(traversableArray[i])  
  
 return listPrimeNumber, len(listPrimeNumber)  
  
  
def SearchForPrimeNumbersOnTheIntervalUsingRecursion(listNumber = []):  
 if (len(listNumber) == 2):  
 listPrimeNumber = []  
 traversableArray = np.arange(listNumber[0], listNumber[1] + 1, 1)  
  
 if (listNumber != [2, 3]):  
 primeNumbersUpToUpperBound = SearchForPrimeNumbersOnTheIntervalUsingRecursion([2, math.ceil(math.pow(listNumber[1], 0.5))])[0]  
 else:  
 primeNumbersUpToUpperBound = [2, 3]  
  
 for i in range(len(traversableArray)):  
 for j in range(len(primeNumbersUpToUpperBound)):  
 if (traversableArray[i] % primeNumbersUpToUpperBound[j] == 0 and traversableArray[i] !=  
 primeNumbersUpToUpperBound[j]):  
 break  
 else:  
 listPrimeNumber.append(traversableArray[i])  
  
 return listPrimeNumber, len(listPrimeNumber)  
  
  
def IsAPrimeNumber(prime\_number):  
 if (prime\_number > 1):  
 for i in range(2, round(math.pow(prime\_number, 0.5))):  
 if (prime\_number % i == 0):  
 print("Число", prime\_number, "не является простым!")  
 is\_prime = False  
 break  
 else:  
 print("Число", prime\_number, "является простым!")  
 is\_prime = True  
 return is\_prime  
  
  
def DecompositionIntoPrimeNumbers(divisible\_number):  
 list\_divider = []  
 np\_number = np.arange(2, divisible\_number + 1)  
  
 if (IsAPrimeNumber(divisible\_number)):  
 list\_divider.append(1)  
 list\_divider.append(divisible\_number)  
 else:  
 for current\_numb in np\_number:  
 if (divisible\_number % current\_numb == 0):  
 list\_divider.append(current\_numb)  
  
 print("Делителями числа", divisible\_number, "являются числа", list\_divider)  
  
  
def DetermineThePrimeFactorsOfANumber(prime\_factors\_number):  
 list\_prime\_divider = []  
 np\_number = SearchForPrimeNumbers(prime\_factors\_number)[0]  
  
 if (IsAPrimeNumber(prime\_factors\_number)):  
 list\_prime\_divider.append(1)  
 list\_prime\_divider.append(prime\_factors\_number)  
 else:  
 while prime\_factors\_number != 1:  
 for current\_numb in np\_number:  
 if (prime\_factors\_number % current\_numb == 0):  
 prime\_factors\_number /= current\_numb  
 list\_prime\_divider.append(current\_numb)  
 break  
 print("Простые множители", list\_prime\_divider)  
  
start\_time = time.time()  
print("Простые числа до определённого числа, используя чистое решето Эратосфена:", SearchForPrimeNumbers(1000))  
print("--- %s seconds ---" % (time.time() - start\_time))  
  
start\_time = time.time()  
print("Простые числа на интервале, дополнительно используя решето Эратосфена:", SearchForPrimeNumbersOnTheInterval([2, 1000])[1])  
print("--- %s seconds ---" % (time.time() - start\_time))  
  
start\_time = time.time()  
print("Простые числа на интервале, дополнительно используя рекурсию:", SearchForPrimeNumbersOnTheIntervalUsingRecursion([2, 1000])[1])  
print("--- %s seconds ---" % (time.time() - start\_time), "\n")  
  
main()

**Результат выполнения**

Простые числа до определённого числа, используя чистое решето Эратосфена: ([2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997], 145, 168)

--- 0.16701316833496094 seconds ---

Простые числа на интервале, дополнительно используя решето Эратосфена: 168

--- 0.0019991397857666016 seconds ---

Простые числа на интервале, дополнительно используя рекурсию: 168

--- 0.0020020008087158203 seconds ---

----- Наибольший общий делитель -----

Список чисел: [140, 420, 280, 140, 140]

НОД чисел: 140

----- Поиск простых чисел -----

Простые числа [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]

Простые числа по формуле n/ln(n): 145

Простые числа действительно: 168

----- Простое ли число??? -----

Число 24 не является простым!

----- Делители числа -----

Число 27 не является простым!

Делителями числа 27 являются числа [3, 9, 27]

----- Разложение на простые множители -----

Число 260 не является простым!

Простые множители [2, 2, 5, 13]

**Ответы на контрольные вопросы**

1. Дать определение понятий: целое число, натуральное число, делимость чисел, собственный делитель, НОД.

Целое число – действительные числа без дробной части.

Натуральное число – целые числа, которые используются при счёте.

Делимость чисел – свойство числа а и b, при котором a= b\*q, где все числа целые.

Собственный делитель – Делитель a называется собственным делителем числа b, если 1 < |a| < |b|.

НОД – Наибольший общий делитель одного или нескольких чисел.

2. Сформулировать основную теорему арифметики. Представить примеры ее применения.

Всякое натуральное число n, кроме 1, можно представить, как произведение простых множителей. 12 = 3\*2\*2.

3. Пояснить сущность проблемы факторизации и ее связь с прикладной криптографией.

Проблема факторизации - сложность решения задачи разложения больших чисел на простые сомножители. Если числа имеют большой размер, то компьютеры могут искать ответ столетиями, что делает системы криптоустойчивами.

4. Найти НОД: пар чисел: 333 и 100; 56 и 200; 99 и 200; 61 и 987; 123 и 456; трех чисел: 21, 43, 342; 57, 31, 200; 42, 11, 98.

НОД(333,100) = 1;

НОД(56,200) = 8;

НОД(99,200) = 1;

НОД(61,987) = 1;

НОД(123,456) = 3;

НОД(21,43,342) = 1;

НОД(57,31,200) = 1;

НОД(42,11,98) = 1;

5. Записать каноническое разложение чисел: 2770; 3780; 6224.

2770 = 2\*5\*277

3780 = 2\*5\*2\*7\*3\*3\*3.

6224 = 2\*2\*2\*2\*389

6. Записать соотношение Безу. Показать пример его практического использования.

Соотношение между парой целых чисел a, b и их наибольшим общим делителем: НОД (a, b) = ax + by.

7. Подсчитать число взаимно простых чисел с числами 2770, 3780, 6224.

2770 = 2770\*(1-1/2)\* (1-1/5)\* (1-1/277) = 1104

3780 = 3780\*(1-1/2)\* (1-1/2)\* (1-1/5) \*(1-1/3)\* (1-1/3)\* (1-1/3) \* (1-1/7) = 192

6224 =6224\*(1-1/2)\* (1-1/2)\* (1-1/2) \*(1-1/2) \*(1-1/389) = 388

8. Сформулировать малую теорему Ферма. Показать примеры ее практического применения.

Если p — простое число и a — целое число, не делящееся на p, то ар-1-1 делится на p.

К примеру, если a=2; p=7, то 26=64, и 64-1=63=7\*9.

**Вывод**: модулярная арифметика применяется в криптографии для того, чтобы было нельзя было эффективно использовать методы подбора.