Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра информатики и веб-дизайна

**Отчет к лабораторной работе**:

«Основы теории чисел и их использование в криптографии»

Выполнила:

студентка 3 курса 9 группы

специальности ДЭИВИ

Змитревич Д. А.

Минск 2023

**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.

1. Теоретические сведения

Теория чисел, или *высшая арифметика*, – раздел математики, изучающий натуральные числа и иные похожие величины.

Множество всех целых чисел есть набор всех действительных чисел без дробной части.

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

Несколько важных свойств простых чисел:

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

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

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.

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.

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

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

2. Практическое задание

Разработанное авторское приложение реализовывает операции по расчету НОД переменного количества чисел и поиску простых чисел в заданном промежутке. Код функций представлен ниже.

*function* searchSimple($start, $end) {

$count = 0;

$numbers = [];

for ($i = $start; $i <= $end; $i++) {

$flag = true;

for ($j = 2; $j < $i; $j++) {

if ($i % $j == 0 && $i % 1 == 0) {

$flag = false;

}

}

if ($flag) {

$numbers[] = $i;

$count++;

}

}

return [$count, implode(', ', $numbers)];

}

Рисунок 1.1 – Функция на поиск простых чисел в промежутке

В качестве параметров передаются два числа. Функция подразумевает собой работу с флагом, который принимает значение «false», если число не является простым, т.е. делится на число отличное от единицы или самого себя, проверка на что происходит в цикле, который перебирает все значения до последнего переданного параметра. Возвращает массив из количества простых чисел и найденные простые числа записанные в строку через запятую.

*function* gcd(...$numbers) {

$result = $numbers[0];

for ($i = 1; $i < count($numbers); $i++) {

$result = gcd\_two\_numbers($result, $numbers[$i]);

}

return $result;

}

*function* gcd\_two\_numbers($a, $b) {

while ($b != 0) {

$t = $b;

$b = $a % $b;

$a = $t;

}

return $a;

}

Рисунок 1.2 – Функция на нахождение НОД переменного количества чисел

В функции используется оператор ... для определения переменного количества аргументов в функции gcd(). Затем мы инициализируем переменную $result с первым переданным аргументом и используем цикл for для прохода по оставшимся аргументам и вычисления НОД для всех чисел. Функция gcd\_two\_numbers() находит НОД для двух чисел, используя алгоритм Евклида.

Вариант 3, m=367, n=401.

**Решение задания №1.**

Найти все простые числа в интервале [2, n], подсчитать их количество и сравнить с n/ln(n).

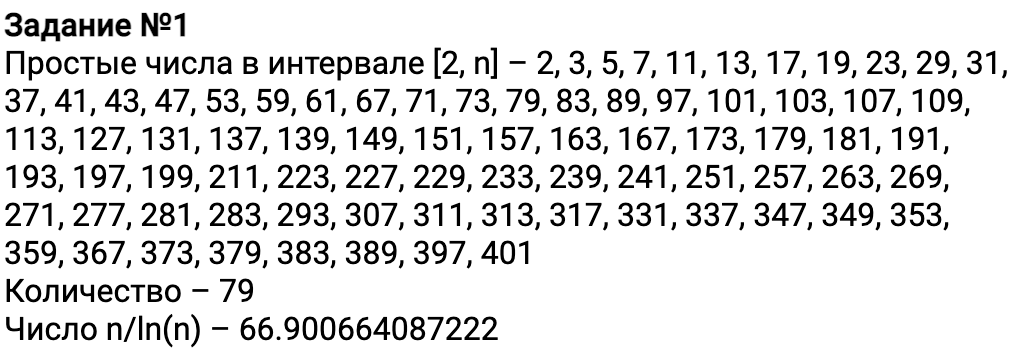


Рисунок 1.3 – Результат решения задания №1

Для решения использовалась функция searchSimple. Согласно свойству 2 простых чисел, вычислим n / ln(n) = 401 / ln(401) ≈ 66,9. Результат почти близок к истинному.

**Решение задания №2.**

Найти все простые числа в интервале [m, n], cравнить полученные результаты с «ручными» вычислениями, используя «решето Эратосфена».

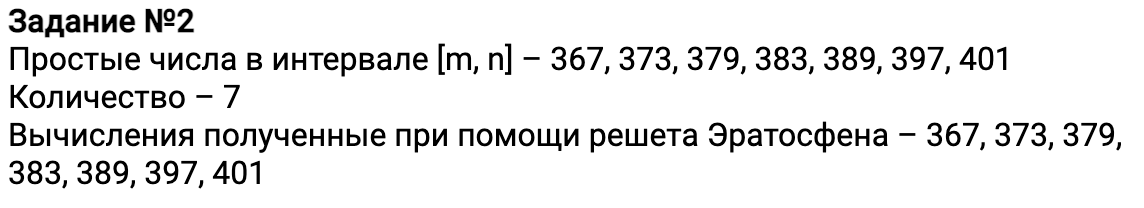


Рисунок 1.4 – Результат решения задания №2

Для поиска простых чисел использовалась функция searchSimple. Для «ручных» вычислений была написана функция sieveOfEratosthenes.

*function* sieveOfEratosthenes($start, $end) {

$numbers = range($start, $end);

for ($i = 2; $i <= sqrt($end); $i++) {

for ($j = $i \* $i; $j <= $end; $j += $i) {

if (in\_array($j, $numbers)) {

unset($numbers[array\_search($j, $numbers)]);

}

}

}

return implode(', ', array\_values($numbers));

}

Рисунок 1.5 – Функция, реализовывающая алгоритм «решето Эратосфена»

Функция работает с массивом, в который записываются все числа в указанном промежутке. Идет перебор всех чисел и во внутреннем цикле из массива убираются числа кратные числу из внешнего цикла, тем самым оставляя лишь простые числа. Результатом работы данной функции являются все простые числа записанные в строку.

**Решение задания №3.**

Записать числа m и n в виде произведения простых множителей (форма записи – каноническая).

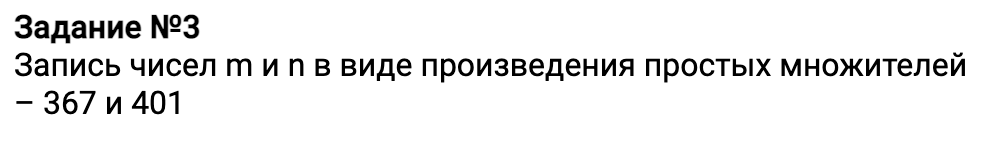


Рисунок 1.5 – Результат решения задания №3

Для решения данной задачи была написана функция primeFactorization.

*function* primeFactorization($n) {

$factors = array();

$divisor = 2;

while ($n >= $divisor) {

if ($n % $divisor == 0) {

$n /= $divisor;

array\_push($factors, $divisor);

} else {

$divisor++;

}

}

$result = implode(' x ', $factors);

return $result;

}

Рисунок 1.6 – Функция, которая раскладывает число на простые множители

Функция работает с массивом, в который поочередно записываются простые множители, если остаток от деления числа на них равен нулю. Числа являются простыми, следовательно, запись разложения их на простые множители будет состоять из одного числа.

**Решение задания №4.**

Проверить, является ли число, состоящее из конкатенации цифр m и n, простым.



Рисунок 1.7 – Результат решения задания №4

Для решения данной задачи была написана функция isPrime.

*function* isPrime($n) {

if ($n <= 1) {

return false;

}

for ($i = 2; $i <= sqrt($n); $i++) {

if ($n % $i == 0) {

return false;

}

}

return true;

Рисунок 1.8 – Функция для проверки является ли число простым

Функция реализует на примере свойство 3 простых чисел.

**Решение задания №5.**

Найти НОД (m, n).

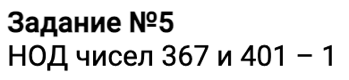


Рисунок 1.9 – Результат решения задания №5

Для решения данной задачи была использована функция gcd. По причине того, что числа являются простыми, их наибольший общий делитель равен единице.

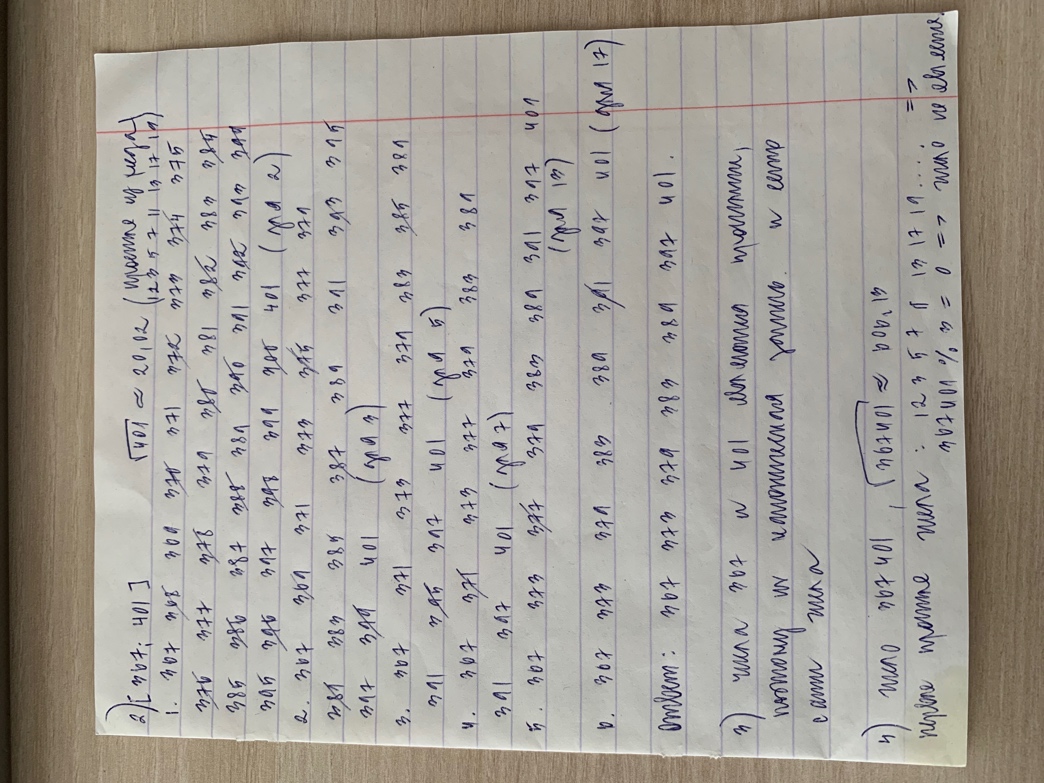


Рисунок 1.10 – Ручной расчет заданий 2, 3 и 4

**Вывод**

В ходе данной лабораторной работы были получены практические навыки нахождения простых чисел в ряду и нахождения наибольшего общего делителя для двух и более чисел.

**Контрольные вопросы**

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

Целое число – это число без дробной части, которое может быть представлено в виде положительного, отрицательного или нуля.

Натуральное число - это число, которое используется для подсчета и представляет собой положительное целое число, начинающееся с единицы и продолжающееся бесконечно.

Делимость чисел - это свойство, по которому одно число может быть равномерно разделено на другое число без остатка.

Собственный делитель - это положительное целое число, которое делит данное число без остатка, кроме самого этого числа.

Наибольшее целое число, которое делит без остатка числа a и b, называется наибольшим общим делителем этих чисел.

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

Основная теорема арифметики утверждает, что любое целое число больше 1 может быть единственным образом представлено как произведение простых чисел в порядке возрастания. Примеры применения этой теоремы включают факторизацию целых чисел на простые множители, нахождение наибольшего общего делителя двух чисел и проверку взаимной простоты двух чисел. Например, для чисел 24 и 36, основная теорема арифметики позволяет представить их в виде произведения простых множителей: 24 = 2^3 × 3 и 36 = 2^2 × 3^2. Нахождение наибольшего общего делителя двух чисел может быть выполнено путем нахождения общих простых множителей их разложений на простые множители. Например, для чисел 24 и 36, НОД равен 2^2 × 3 = 12.

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

Факторизация – это процесс разложения целого числа на простые множители. Эта проблема имеет вычислительную сложность и может быть трудной для больших чисел, особенно если эти числа имеют большие простые множители. В прикладной криптографии проблема факторизации используется для создания криптографических алгоритмов, к примеру RSA, который основан на трудности факторизации больших целых чисел. Криптографические ключи генерируются путем выбора двух больших простых чисел и их перемножения, а затем факторизация такого числа становится сложной задачей для злоумышленников, которые пытаются раскрыть данные, зашифрованные с использованием такого ключа.

4. Найти НОД, пар чисел:

333 и 100 = 1

56 и 200 = 8

99 и 200 = 1

61 и 987 = 1

123 и 456 = 3

трех чисел:

21, 43, 342 = 1

57, 31, 200 = 1

42, 11, 98 = 1

5. Записать каноническое разложение чисел:

2770 = 2 × 5 × 277

3780 = 2 × 2 × 3 × 3 × 3 × 5 × 7

6224 = 2 × 2 × 2 × 2 × 389

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

Если НОД (a, b) = d, то справедливо следующее соотношение (соотношение Безу):

аu + bv = d.

Одним из практических применений соотношения Безу является нахождение наибольшего общего делителя двух чисел.

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

С помощью формулы Эйлера можно подсчитать количество чисел взаимно простых с приведенным. Для подсчета была написана функция, которая ее реализует.

*function* euler($n) {

$result = $n;

for ($i = 2; $i <= sqrt($n); $i++) {

if ($n % $i == 0) {

while ($n % $i == 0) {

$n /= $i;

}

$result -= $result / $i;

}

} if ($n > 1) {

$result -= $result / $n;

}

return $result;

}

Рисунок 1.10 – Реализация функции Эйлера

Тем самым число взаимно простых чисел с числами,

2770 = 1104,

3780 = 864,

6224 = 3104

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

Малая теорема Ферма гласит: если p – простое число, а a не делится на p, то a^(p-1) - 1 делится на p.

Малая теорема Ферма имеет много практических применений, особенно в криптографии и теории чисел. Например, она используется в RSA-шифровании для проверки корректности выбора больших простых чисел.

9. Сформулировать основные свойства модулярной арифметики.

Модулярная арифметика — это арифметика, которая проводится по модулю некоторого числа. Основные свойства модулярной арифметики:

– Коммутативность: a + b = b + a, a \* b = b \* a в модульной арифметике.

– Ассоциативность: (a + b) + c = a + (b + c), (a \* b) \* c = a \* (b \* c) в модульной арифметике.

– Распределительный закон: a \* (b + c) = (a \* b) + (a \* c) в модульной арифметике.

– Остаток от суммы: (a + b) % m = ((a % m) + (b % m)) % m в модульной арифметике.

– Остаток от произведения: (a \* b) % m = ((a % m) \* (b % m)) % m в модульной арифметике.

– Обратный элемент: если a и m взаимно просты, то существует такое число b, что (a \* b) % m = 1.

10. Пояснить порядок операций на основе расширенного алгоритма Евклида.

Расширенный алгоритм Евклида – это метод для нахождения наибольшего общего делителя двух целых чисел и одновременного нахождения их линейной комбинации, которая равна этому НОДу. Он выполняется следующим образом:

– Пусть даны два числа a и b, где a ≥ b.

– Используя обычный алгоритм Евклида, найдем их НОД. Для этого необходимо последовательно вычислять остатки от деления a на b, затем b на остаток и т.д., пока остаток не станет равным нулю. НОД будет равен последнему ненулевому остатку.

– Вычислим коэффициенты x и y для линейной комбинации a и b, используя следующие формулы:

x0 = 1, y0 = 0

x1 = 0, y1 = 1

xn = xn-2 - qxn-1, yn = yn-2 - qyn-1, где q – целая часть от деления xn-2 на xn-1, и n ≥ 2.

Когда остаток станет равным нулю, возвращаем коэффициенты x и y для предыдущей итерации (которая была последней с ненулевым остатком), они будут равным x и y.

– На выходе получаем НОД(a, b) и коэффициенты x и y, такие что ax + by = НОД.

11. Найти числа, обратные к а по модулю n:

a = 41, n = 143;

a = 13, n = 71.

Для того, чтобы найти числа, обратные к числу a по модулю n, необходимо решить уравнение ax ≡ 1 (mod n).

Для первого случая, где a = 41 и n = 143:

Используя расширенный алгоритм Евклида, находим НОД(41,143) и коэффициенты Безу, которые удовлетворяют уравнению 41x + 143y = НОД(41,143):

НОД(41,143) = 1

коэффициент x = -23

коэффициент y = 7

Полученный коэффициент x является числом, обратным к a = 41 по модулю n = 143. Однако, так как значение коэффициента отрицательное, необходимо добавить значение n к нему:

x = -23 + 143 = 120

Таким образом, число 120 является числом, обратным к a = 41 по модулю n = 143.

Для второго случая, где a = 13 и n = 71:

Используя расширенный алгоритм Евклида, находим НОД(13,71) и коэффициенты Безу, которые удовлетворяют уравнению 13x + 71y = НОД(13,71):

НОД(13,71) = 1

коэффициент x = -18

коэффициент y = 3

Полученный коэффициент x является числом, обратным к a = 13 по модулю n = 71. Однако, так как значение коэффициента отрицательное, необходимо добавить значение n к нему:

x = -18 + 71 = 53

Таким образом, число 53 является числом, обратным к a = 13 по модулю n = 71.