Донецкий Национальный Технический Университет

Лабораторная работа № 2

«АлгоритмЫ ВЫЧИСЛЕНИЯ НОК И НОД»

Выполнил:

ст. группы ИПЗ -13

Лысенко А. С.

Проверила:

Скрипник Т.В.

Красноармейск 2015

Введение

Наибольший общий делитель (НОД) n данных чисел - это наибольшее число, на которое n данных чисел делятся без остатка.

Наименьшее общее кратное (НОК) n натуральных чисел называется наименьшее натуральное число, которое само делится нацело на каждое из этих чисел.

Алгоритм Евклида — эффективный [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) для нахождения [наибольшего общего делителя](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D0%B8%D0%B1%D0%BE%D0%BB%D1%8C%D1%88%D0%B8%D0%B9_%D0%BE%D0%B1%D1%89%D0%B8%D0%B9_%D0%B4%D0%B5%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C) двух [целых чисел](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE). В самом простом случае алгоритм Евклида применяется к паре положительных целых чисел и формирует новую пару, которая состоит из меньшего числа и [разницы](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%82%D0%B0%D0%BD%D0%B8%D0%B5) между большим и меньшим числом. Процесс повторяется, пока числа не станут равными. Найденное число и есть наибольший общий делитель исходной пары.



Иначе говоря, НОД двух натуральных чисел равен НОД их положительной разности (модуля их разности) и меньшего числа.



Одним из методов ускорения целочисленного алгоритма Евклида является использование симметричного остатка.

Программная реализация

**package** Main;  
  
**import** java.io.BufferedReader;  
**import** java.io.IOException;  
**import** java.io.InputStreamReader;  
**import** java.io.Reader;  
  
**public class** Source  
{  
 **public static void** main(String[] args)**throws** IOException  
 {  
 **new** Source();  
 }  
  
 **private** Source() **throws** IOException  
 {  
 **this**.GCD();  
 System.***out***.println();  
 **this**.gcd();  
 }  
 **private int** GCD() **throws** IOException  
 {  
 *// long start = System.currentTimeMillis();  
 // String a = "";  
 //String b = "";* **int** a1 = 900;  
 **int** b1 = 1387;  
 **int** count =0 ;  
 **int** c1;  
 *// BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));* System.***out***.println(**"a : "** + a1);  
 *// a = reader.readLine();  
 //a1 = Integer.valueOf(a).intValue();* System.***out***.println(**"b : "** + b1);  
 *// b = reader.readLine();  
 // b1 = Integer.valueOf(b).intValue();* c1 = a1 \* b1;  
 **while**(a1 != b1)  
 {  
 **if** (a1 > b1)  
 a1 -= b1;  
 **else** b1 -=a1;  
 count++;  
 }  
 c1 = c1 / a1;  
 *//long finish = System.currentTimeMillis();  
 // long time = finish - start;* System.***out***.println(**"НОД : "** + a1);  
 System.***out***.println(**"НОК (a \* b / НОД): "** + c1);  
 System.***out***.println(**"Кол-во операций : "** + count);  
 *// System.out.println("Время : " + time);* **return** a1;  
 }  
 **private int** gcd() **throws** IOException  
 {  
 *//long start1 = System.currentTimeMillis();  
 //String a = "";  
 //String b = "";* **int** a = 900;  
 **int** b = 1387;  
 **int** count = 0;  
 **int** c = 0;  
 *// BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));* System.***out***.println(**"a : "** + a);  
 *// a = reader.readLine();  
 // a = Integer.valueOf(a).intValue();* System.***out***.println(**"b : "** + b);  
 *// b = reader.readLine();  
 // b = Integer.valueOf(b).intValue();* c = a \* b;  
 **while** (b !=0)  
 {  
 **int** tmp = a%b;  
 a = b;  
 b = tmp;  
 count++;  
 }  
 c = c / a;  
 *//long finish1 = System.currentTimeMillis();  
 //long time1 = finish1 - start1;* System.***out***.println(**"НОД : "** + a);  
 System.***out***.println(**"НОК (a \* b / НОД): "** + c);  
 System.***out***.println(**"Кол-во операций : "** + count);  
 *//System.out.println("Время : " + time1);* **return** a;  
 }  
}

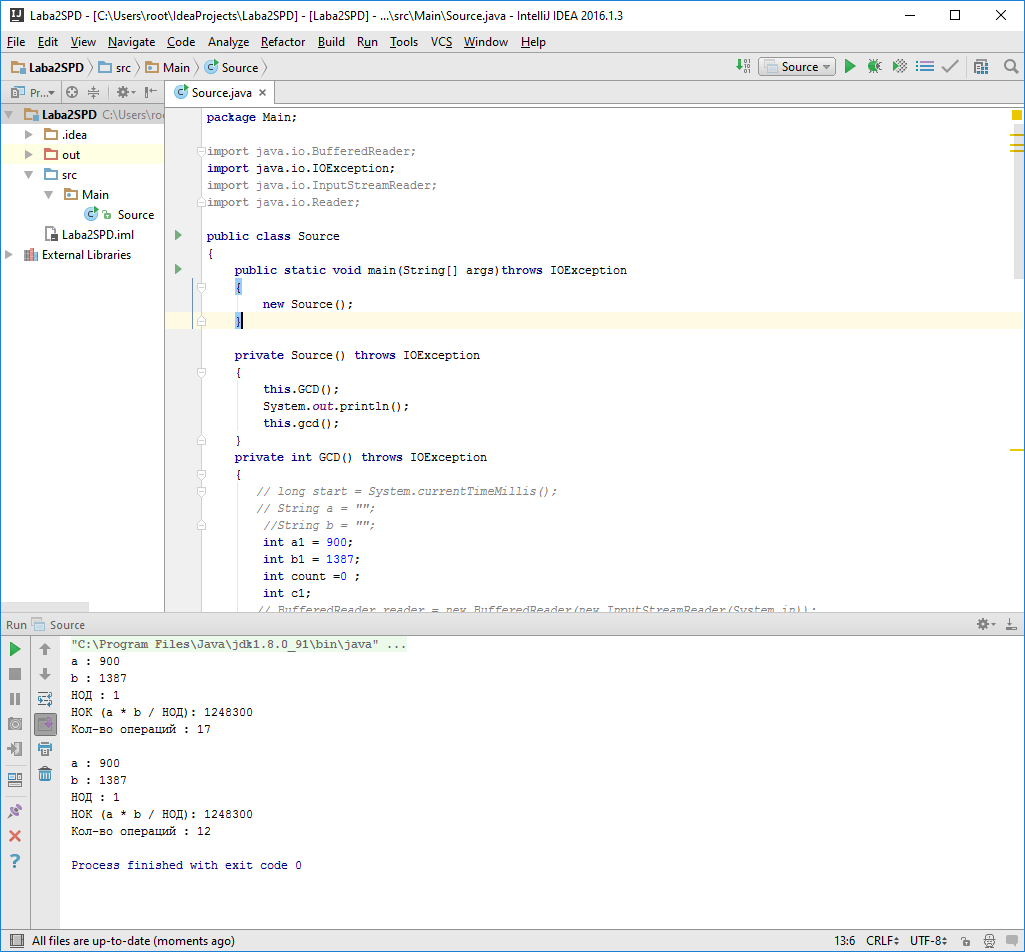


Рисунок 1

Вывод

Алгоритм симметричного остатка работает быстрее на 5 операций.