# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

### Факультет физико-математических и естественных наук

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4 =============== ## Тема: Вычисление наибольшего общего делителя дисциплина: Математические основы защиты информации и информационной безопасности

Студент: Койфман Кирилл Дмитриевич

Группа: НФИмд-01-25

## Введение

### Цель работы

Получение практических навыков реализации алгоритмов, вычисляющих наибольший общий делитель (НОД).

### Задачи

1. Реализовать алгоритм Евклида, бинарный алгоритм Евклида, расширенный алгоритм Евклида, расширенный бинарный алгоритм Евклида.

## Теория

Пусть числа и целые и . Разделить на с остатком - значит предоставить в виде , где и . Число называется неполным частным, число - неполным остатком от деления на .

Целое число называется *наибольшим общим делителем* целых чисел (обозначается ), если выполняются следующие условия: 1. каждое из чисел делится на ; 2. если - другой общий делитель чисел , то делится на . Например, , , .

Ненулевые целые числа и называются *ассоциированными* (обозначается ~), если делится на и делится на .

Для любых целых чисел существует наибольший общий делитель и его можно представить в виде *линейной комбинации* этих чисел:

Например, НОД чисел 91, 105, 154 равен 7. В качестве линейного представления можно взять

либо

Целые числа называются *взаимно простыми в совокупности*, если . Целые числа и называются *взаимно простыми*, если .

Целые числа , называются *попарно взаимно простыми*, если для всех .

## ## Ход работы

Для решения поставленной задачи реализуем описанные в тексте лабораторной работы алгоритмы для вычисления наибольшего общего делителя (НОД) на языке программирования C++ (Листинг-1 - Листинг-4), а также проведём тест данных алгоритмов, чтобы проверить корректность их работы (Листинг-5):

int algorithmEuclid(int a, int b)  
{  
 if (b > a)  
 std::swap(a, b);  
   
 //d=НОД(a, b)  
 int d = 0;  
  
 int gamma\_prev = a;  
 int gamma\_curr = b;  
 int gamma\_next = 0;  
 while (true)  
 {  
 int remainder = gamma\_prev % gamma\_curr;  
 gamma\_next = remainder;  
  
 if (gamma\_next == 0)  
 {  
 d = gamma\_curr;  
 break;  
 }  
 else  
 {  
 gamma\_prev = gamma\_curr;  
 gamma\_curr = gamma\_next;  
 }  
 }  
  
 return d;  
}

*Листинг-1(реализация алгоритма Евклида)*

int binaryAlgorithmEuclid(int a, int b)  
{  
 if (b > a)  
 std::swap(a, b);  
   
 //d=НОД(a, b)  
 int d = 0;  
  
 int g = 1;  
 //until one in pair (a or b) becomes odd  
 while (a % 2 == 0 && b % 2 == 0)  
 {  
 a = a / 2;  
 b = b / 2;  
 g = 2 \* g;  
 }  
  
 int u = a;  
 int v = b;  
 while (u != 0)  
 {  
 if (u % 2 == 0)  
 u = u / 2;  
  
 if (v % 2 == 0)  
 v = v / 2;  
  
 if (u >= v)  
 u = u - v;  
 else  
 v = v - u;  
 }  
 d = g \* v;  
  
 return d;  
}

*Листинг-2(реализация бинарного алгоритма Евклида)*

struct EuclidAlgoVars  
{  
 int d, x, y;  
};  
  
EuclidAlgoVars extendedAlgorithmEuclid(int a, int b)  
{  
 bool swapXY = false;  
 if (b > a)  
 {  
 swapXY = true;  
 std::swap(a, b);  
 }  
   
 //a \* x + b \* y = d  
 int d = 0;  
 int x = 0;  
 int y = 0;  
  
 //gamma\_0{i-1}  
 int gamma\_prev = a;  
 //gamma\_1{i}  
 int gamma\_curr = b;  
 //gamma\_{i+1}  
 int gamma\_next = 0;  
  
 //x\_0{i-1}  
 int x\_prev = 1;  
 //x\_1{i}  
 int x\_curr = 0;  
 //x\_{i+1}  
 int x\_next = 0;  
  
 //y\_0{i-1}  
 int y\_prev = 0;  
 //y\_1{i}  
 int y\_curr = 1;  
 //y\_{i+1}  
 int y\_next = 0;  
  
 while (true)  
 {  
 int remainder = gamma\_prev / gamma\_curr;  
 //q\_i = remainder  
 int q\_curr = remainder;  
 //gamma\_{i+1} = gamma\_{i-1} - q\_i \* gamma\_i  
 gamma\_next = gamma\_prev - q\_curr \* gamma\_curr;  
  
 if (gamma\_next == 0)  
 {  
 d = gamma\_curr;  
 x = x\_curr;  
 y = y\_curr;  
 break;  
 }  
 else  
 {  
 x\_next = x\_prev - q\_curr \* x\_curr;  
 y\_next = y\_prev - q\_curr \* y\_curr;  
  
 gamma\_prev = gamma\_curr;  
 gamma\_curr = gamma\_next;  
  
 x\_prev = x\_curr;  
 x\_curr = x\_next;  
  
 y\_prev = y\_curr;  
 y\_curr = y\_next;  
 }  
 }  
  
 if (swapXY)  
 std::swap(x, y);  
   
 return EuclidAlgoVars{ d,x,y };  
}

*Листинг-3(реализация расширенного алгоритма Евклида)*

EuclidAlgoVars binaryExtendedAlgorithmEuclid(int a, int b)  
{  
 bool swapXY = false;  
 if (b > a)  
 {  
 swapXY = true;  
 std::swap(a, b);  
 }  
 //a \* x + b \* y = d  
 int d = 0;  
 int x = 0;  
 int y = 0;  
  
 int g = 1;  
 //until one in pair (a or b) becomes odd  
 while (a % 2 == 0 && b % 2 == 0)  
 {  
 a = a / 2;  
 b = b / 2;  
 g = 2 \* g;  
 }  
  
 int u = a;  
 int v = b;  
 int A = 1;  
 int B = 0;  
 int C = 0;  
 int D = 1;  
  
 while (u != 0)  
 {  
 if (u % 2 == 0)  
 {  
 u = u / 2;  
 if (A % 2 == 0 && B % 2 == 0)  
 {  
 A = A / 2;  
 B = B / 2;  
 }  
 else  
 {  
 A = (A + b) / 2;  
 B = (B - a) / 2;  
 }  
 }  
  
 if (v % 2 == 0)  
 {  
 v = v / 2;  
 if (C % 2 == 0 && D % 2 == 0)  
 {  
 C = C / 2;  
 D = D / 2;  
 }  
 else  
 {  
 C = (C + b) / 2;  
 D = (D - a) / 2;  
 }  
 }  
  
 if (u >= v)  
 {  
 u = u - v;  
 A = A - C;  
 B = B - D;  
 }  
 else  
 {  
 v = v - u;  
 C = C - A;  
 D = D - B;  
 }  
   
 }  
 d = g \* v;  
 x = C;  
 y = D;  
  
 if (swapXY)  
 std::swap(x, y);  
   
 return EuclidAlgoVars{ d,x,y };  
}

*Листинг-4(реализация расширенного бинарного алгоритма Евклида)*

--------------------------Testing GCD-algorithms--------------------------  
TEST-1: a = 14, b = 21  
Great Common Divisor(GCD) for pair{a=14, b=21} with [1]Euclid Algorithm: 7  
Great Common Divisor(GCD) for pair{a=14, b=21} with [2]Binary Euclid Algorithm: 7  
Great Common Divisor(GCD) for pair{a=14, b=21} with [3]Extended Euclid Algorithm: 7 with condition that a \* x + b \* y = d, d = 7, x = -1, y = 1:  
14 \* -1 + 21 \* 1 = 7(true)  
Great Common Divisor(GCD) for pair{a=14, b=21} with [4]Binary Extended Euclid Algorithm: 7 with condition that a \* x + b \* y = d, d = 7, x = -10, y = 7:  
14 \* -10 + 21 \* 7 = 7(true)  
  
TEST-2: a = 48, b = 36  
Great Common Divisor(GCD) for pair{a=48, b=36} with [1]Euclid Algorithm: 12  
Great Common Divisor(GCD) for pair{a=48, b=36} with [2]Binary Euclid Algorithm: 12  
Great Common Divisor(GCD) for pair{a=48, b=36} with [3]Extended Euclid Algorithm: 12 with condition that a \* x + b \* y = d, d = 12, x = 1, y = -1:  
48 \* 1 + 36 \* -1 = 12(true)  
Great Common Divisor(GCD) for pair{a=48, b=36} with [4]Binary Extended Euclid Algorithm: 12 with condition that a \* x + b \* y = d, d = 12, x = -5, y = 7:  
48 \* -5 + 36 \* 7 = 12(true)  
  
TEST-3: a = 17, b = 51  
Great Common Divisor(GCD) for pair{a=17, b=51} with [1]Euclid Algorithm: 17  
Great Common Divisor(GCD) for pair{a=17, b=51} with [2]Binary Euclid Algorithm: 17  
Great Common Divisor(GCD) for pair{a=17, b=51} with [3]Extended Euclid Algorithm: 17 with condition that a \* x + b \* y = d, d = 17, x = 1, y = 0:  
17 \* 1 + 51 \* 0 = 17(true)  
Great Common Divisor(GCD) for pair{a=17, b=51} with [4]Binary Extended Euclid Algorithm: 17 with condition that a \* x + b \* y = d, d = 17, x = 1, y = 0:  
17 \* 1 + 51 \* 0 = 17(true)  
  
TEST-4: a = 75, b = 250  
Great Common Divisor(GCD) for pair{a=75, b=250} with [1]Euclid Algorithm: 25  
Great Common Divisor(GCD) for pair{a=75, b=250} with [2]Binary Euclid Algorithm: 25  
Great Common Divisor(GCD) for pair{a=75, b=250} with [3]Extended Euclid Algorithm: 25 with condition that a \* x + b \* y = d, d = 25, x = -3, y = 1:  
75 \* -3 + 250 \* 1 = 25(true)  
Great Common Divisor(GCD) for pair{a=75, b=250} with [4]Binary Extended Euclid Algorithm: 25 with condition that a \* x + b \* y = d, d = 25, x = -93, y = 28:  
75 \* -93 + 250 \* 28 = 25(true)  
  
TEST-5: a = 72, b = 120  
Great Common Divisor(GCD) for pair{a=72, b=120} with [1]Euclid Algorithm: 24  
Great Common Divisor(GCD) for pair{a=72, b=120} with [2]Binary Euclid Algorithm: 24  
Great Common Divisor(GCD) for pair{a=72, b=120} with [3]Extended Euclid Algorithm: 24 with condition that a \* x + b \* y = d, d = 24, x = 2, y = -1:  
72 \* 2 + 120 \* -1 = 24(true)  
Great Common Divisor(GCD) for pair{a=72, b=120} with [4]Binary Extended Euclid Algorithm: 24 with condition that a \* x + b \* y = d, d = 24, x = -3, y = 2:  
72 \* -3 + 120 \* 2 = 24(true)  
  
TEST-6: a = 81, b = 65  
Great Common Divisor(GCD) for pair{a=81, b=65} with [1]Euclid Algorithm: 1  
Great Common Divisor(GCD) for pair{a=81, b=65} with [2]Binary Euclid Algorithm: 1  
Great Common Divisor(GCD) for pair{a=81, b=65} with [3]Extended Euclid Algorithm: 1 with condition that a \* x + b \* y = d, d = 1, x = -4, y = 5:  
81 \* -4 + 65 \* 5 = 1(true)  
Great Common Divisor(GCD) for pair{a=81, b=65} with [4]Binary Extended Euclid Algorithm: 1 with condition that a \* x + b \* y = d, d = 1, x = -134, y = 167:  
81 \* -134 + 65 \* 167 = 1(true)

*Листинг-5(результаты работы алгоритмов, вычисляющих НОД)*

Исходя из полученных результатов (Листинг-5), можно судить о том, что реализованные алгоритмы успешно вычисляют НОД для пар целочисленных значений.

## Заключение

В ходе проделанной лабораторной работы мной были получены навыки по реализации алгоритмов, вычисляющих НОД.