Федеральное государственное образовательное бюджетное учреждение высшего образования

«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ

РОССИЙСКОЙ ФЕДЕРАЦИИ»

(Финансовый университет)

**Колледж информатики и программирования**

Отчёт о проделанной работе  
по практическому занятию № 1 по дисциплине

МДК.02.02 «Криптографические средства и методы защиты информации»

На тему: «Применение алгоритма Евклида для нахождения НОД. Решение линейных диофантовых уравнений»

Студент группы 3ОИБАС-1221

|  |  |
| --- | --- |
| Верстунин П.В.  Кован Н. В.  Осадчий И. А. | «6» ноября .2023 г. |

Основная профессиональная образовательная программа по специальности

10.02.05 Обеспечение информационной безопасности автоматизированных систем

Форма обучения очная

Проверили:

Рой А.В.,

Москва

2023

**Введение:**

В области математики существует множество интересных задач, связанных с нахождением наименьшего общего делителя (НОД) для пары чисел и решением линейных диофантовых уравнений. Эти задачи имеют важное значение в различных математических и практических областях, таких как криптография, теория чисел и оптимизация. Для успешного решения таких задач широко используется алгоритм Евклида, который представляет собой мощный инструмент для определения НОД и разработки решений линейных диофантовых уравнений. В данной практической работе мы более подробно рассмотрим алгоритм Евклида и его применение в контексте решения описанных математических задач.

**Цель работы:**

* Напишите программу вычисления НОД
* Напишите программу вычисления частных корней линейного диофантова уравнения

**Теоретическая часть:**

**Алгоритм Евклида**

Алгоритм Евклида — эффективный алгоритм для нахождения наибольшего общего делителя двух целых чисел (или общей меры двух отрезков). Алгоритм назван в честь греческого математика Евклида (III век до н. э.), который впервые описал его в VII и X книгах «Начал». Это один из старейших численных алгоритмов, используемых в наше время.

В самом простом случае алгоритм Евклида применяется к паре положительных целых чисел и формирует новую пару, которая состоит из меньшего числа и разницы между большим и меньшим числом. Процесс повторяется, пока числа не станут равными. Найденное число и есть наибольший общий делитель исходной пары. Евклид предложил алгоритм только для натуральных чисел и геометрических величин (длин, площадей, объёмов).

Для данного алгоритма существует множество теоретических и практических применений. В частности, он является основой для криптографического алгоритма с открытым ключом RSA, широко распространённого в электронной коммерции. Также алгоритм используется при решении линейных диофантовых уравнений.

Алгоритм Евклида основан на следующих двух фактах (доказательство см. в приложении Q):

Факт 1: НОД (a, 0) = a

Факт 2: НОД (a, b) = НОД (b, r), где r — остаток от деления a на b

Первый факт говорит, что если второе целое число — 0, наибольший общий делитель равен первому числу. Второй факт позволяет нам изменять значение a на b, пока b не станет 0.

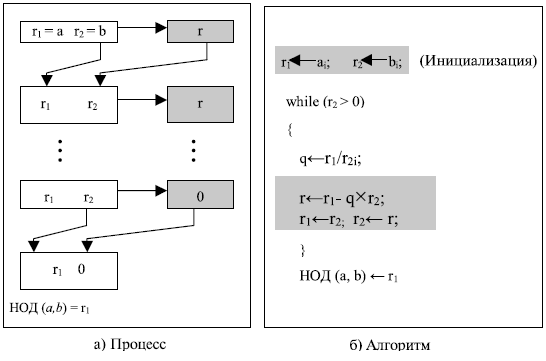


Рис.1. Алгоритм Евклида

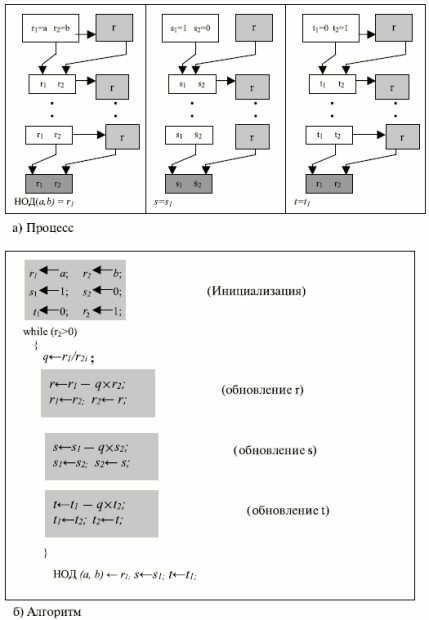
**Расширенный алгоритм Евклида**

Даны два целых числа a и b. Нам зачастую надо найти другие два целых числа, s и t, такие, которые

s x a + t x b = НОД(a,b)

Расширенный алгоритм Евклида может вычислить НОД (a, b) и в то же самое время вычислить значения s и t. Алгоритм и процесс такого вычисления показан на рис. 2.8.

Здесь расширенный алгоритм Евклида использует те же самые шаги, что и простой алгоритм Евклида. Однако в каждом шаге мы применяем три группы вычислений вместо одной. Алгоритм использует три набора переменных: r, s и t.



**Рис. 2.8.** Расширенный алгоритм Евклида

На каждом шаге переменные r1, r2 и r используются так же, как в алгоритме Евклида. Переменным r1 и r2присваиваются начальные значения a и b соответственно. Переменным s1 и s2 присваиваются начальные значения 1 и 0соответственно. Переменным t1 и t2 присваиваются начальные значения 0 и 1, соответственно. Вычисления r, s и tодинаковы, но с одним отличием. Хотя r — остаток от деления r1 на r2, такого соответствия в других двух группах вычислений нет. Есть только одно частное, q, которое вычисляется как r1/r2 и используется для других двух вычислений.

Диофантовыми уравнениями называют уравнения с целыми коэффициентами, для которых требуется найти целочисленные (или натуральные) решения. При этом количество неизвестных в уравнении должно быть не менее двух. Своё название уравнения получили в честь выдающегося античного математика Диофанта Александрийского, который, как считается, первым систематически изучал неопределённые уравнения и описывал методы их решения. Все сохранившиеся записи собраны в книгу «Арифметика».

В частности, линейное диофантово уравнение с двумя неизвестными имеет вид: ax + by = c

**Линейные диофантовы уравнения**

Хотя очень важное приложение расширенного алгоритма Евклида будет рассмотрено далее, здесь мы остановимся на другом приложении — "нахождение решения линейных диофантовых уравнений двух переменных", а именно, уравнения ax + by = c. Мы должны найти значения целых чисел для x и y, которые удовлетворяют этому уравнению. Этот тип уравнения либо не имеет решений, либо имеет бесконечное число решений. Пусть d = НОД (a, b). Если d†c, то уравнение не имеет решения. Если d|c, то мы имеем бесконечное число решений. Одно из них называется частным, остальные — общими.

*Частное решение*

Если d|c, то можно найти частное решение вышеупомянутого уравнения, используя следующие шаги.

* Преобразуем уравнение к a1x + b1y = c1, разделив обе части уравнения на d. Это возможно, потому что d делит a, b, и c в соответствии с предположением.
* Найти s и t в равенстве a1s + b1t = 1, используя расширенный алгоритм Евклида.
* Частное решение может быть найдено:

Частное решение: X0 = (c/d)s и y0 = (c/d)t

*Общие решения*

После нахождения частного решения общие решения могут быть найдены:

Общие решения: x = x0 + k(b/d) и y = y0 – k(a/d), где k — целое число

**Практическая часть:**

**Задание 1. Применение алгоритма Евклида для нахождения НОД.**

1.1. С помощью алгоритма Евклида найдите НОД (дата занятия в формате ГГГГ-ММ-ДД, дата своего рождения ГГГГ-ММ-ДД). Подробно запишите процесс вычисления в редакторе электронных таблиц, поместив таблицу (снимок экрана) в отчёт.

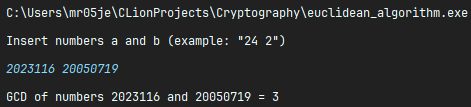
#include <iostream>

#define **cout**(text) std::cout << text << std::endl  
#define **coutd**(text, data) std::cout << text << " = " << data << std::endl;  
#define **cexit**(text) {**cout**(text); *return* 0;}

*int* GCD(*int* a, *int* b);

*int* main() {  
 *int* a, b;  
 **cout**("Insert numbers a and b (example: \"24 2\")");  
 std::cin >> a >> b;  
 std::cout << "GCD of numbers " << a << " and " << b << " = ";  
 **cexit**(GCD(a, b));

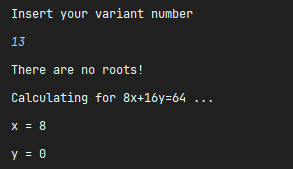
*return* 0;  
}

Результат выполнения программы:  


1.2. Напишите программу вычисления НОД, сравнив результаты с пунктом 1.1. Проверьте работоспособность ещё на нескольких произвольных примерах. Поместите в отчёт исходный текст работающей версии программы и необходимые копии экрана или консоли.

**Задание 2. Решение линейных диофантовых уравнений**

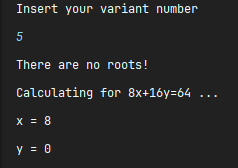
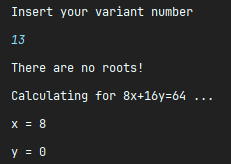
2.1. С помощью алгоритма Евклида найдите частные корни линейного диофантова уравнения a\*x + b\*y = c, где a = (n+1)\*(n+2), b=(n+2)\*(n+3), c=(n+2)^3, n - номер по списку в журнале. Если Вы докажите, что решения для Вашего варианта не существует, то решите Диофантово уравнение 8х+16у=64. Подробно запишите процесс вычисления в редакторе электронных таблиц, поместив таблицу (снимок экрана) в отчёт.

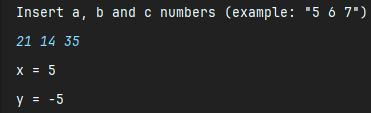


2.2. Напишите программу вычисления частных корней линейного диофантова уравнения из пункта 2.1, сравнив результаты. Проверьте работоспособность ещё на нескольких произвольных примерах. Поместите в отчёт исходный текст работающей версии программы и необходимые копии экрана или консоли.

#include <iostream>  
#include <tuple>  
  
#define **cout**(text) std::cout << text << std::endl  
#define **coutd**(text, data) std::cout << text << " = " << data << std::endl;  
#define **cexit**(text) {**cout**(text); *return* 0;}  
  
*int* GCD(*int* a, *int* b);  
std::tuple<*int*, *int*, *int*> GCDExtended(*int* a, *int* b);  
std::tuple<*int*, *int*> DE(*int* a, *int* b, *int* c);  
  
*int* main() {  
 *int* n, a1, b1, c1;  
  
 **cout**("Insert your variant number");  
 std::cin >> n;  
  
 a1 = (n+1)\*(n+2);  
 b1 = (n+2)\*(n+3);  
 c1 = (n + 2) \* (n + 2) \* (n + 2);  
  
 *auto* de = DE(a1, b1, c1);  
 *int* x = std::get<0>(de);  
 *int* y = std::get<1>(de);  
  
 *if* (x == 0 && y == 0) {  
 **cout**("There are no roots!");  
 **cout**("Calculating for 8х+16у=64 ...");  
  
 a1 = 8; b1 = 16, c1 = 64;  
  
 de = DE(a1, b1, c1);  
 *int* x = std::get<0>(de);  
 *int* y = std::get<1>(de);  
  
 *if* (x == 0 && y == 0) {  
 **cexit**("There are no roots!");  
 }  
 **coutd**("x", x);  
 **coutd**("y", y);  
 } *else* {  
 **coutd**("x", x);  
 **coutd**("y", y);  
 }  
  
 *return* 0;  
}

*int* GCD(*int* a, *int* b) {  
 *return* b == 0 ? a : GCD(b, a % b);  
}  
  
std::tuple<*int*, *int*, *int*> GCDExtended(*int* a, *int* b) {  
 *if* (b == 0) {  
 *return* std::make\_tuple(a, 1, 0);  
 } *else* {  
 *int* gcd, x1, y1;  
 std::tie(gcd, x1, y1) = GCDExtended(b, a % b);  
 *int* x = y1;  
 *int* y = x1 - (a / b) \* y1;  
 *return* std::make\_tuple(gcd, x, y);  
 }  
}  
  
std::tuple<*int*, *int*> DE(*int* a, *int* b, *int* c) {  
 *int* g = GCD(a, b);  
  
 *if* (c % g != 0) *// check if c divides GCD(a, b)  
 return* std::make\_tuple(0, 0);  
  
 a /= g;  
 b /= g;  
 c /= g;  
  
 *auto* gcd\_extended = GCDExtended(a, b);  
  
 *int* gcd = std::get<0>(gcd\_extended);  
 *int* x = std::get<1>(gcd\_extended);  
 *int* y = std::get<2>(gcd\_extended);  
  
 *if* (gcd != 1) {  
 *// If the GCD of a and b is not 1, then there's no solution  
 return* std::make\_tuple(0, 0);  
 }  
  
 x \*= c;  
 y \*= c;  
  
 *return* std::make\_tuple(x, y);  
}



****

**Заключение:**

В данной работе мы изучили и научились вычислять НОД, расширенный НОД, а также решать диофантовые уравнения