**Лабораторная Pабота No 4**

**Математические Основы Защиты Информации и Информационной Безопасности**

Хосе Фернандо Леон Атупанья | НФИмд-01-24 **Содержание**

1. Цель работы
2. Выполнение лабораторной работы
3. Выводы

# Цель работы

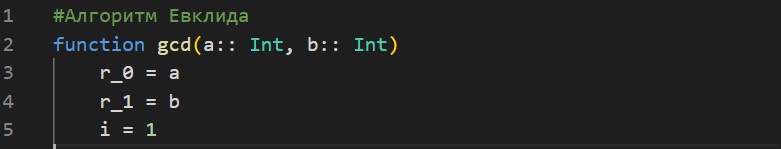
Вычислить максимальный общий делитель, используя алгоритмы, представленные в лабораторном рабочем материале 4.

# Выполнение лабораторной работы

## Алгоритм Евклида

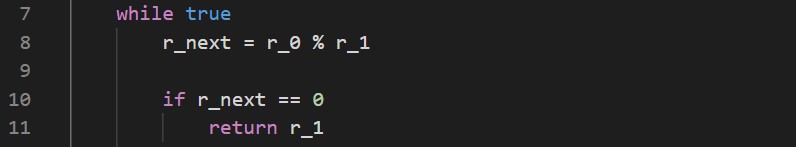
В этой отчете описывается программная реализация евклида алгоритма для нахождения наибольшего общего делителя (НОД) двух чисел. Этот алгоритм, написанный на языке программирования Julia, эффективен при вычислении НОД путем итеравтиного применения операций по модулю.

Функция: Реализация алгоритма Евклида Функция gcd вычисляет GCD из двух целых чисел, используя алгоритм Евклида. Вот краткое описание функции:

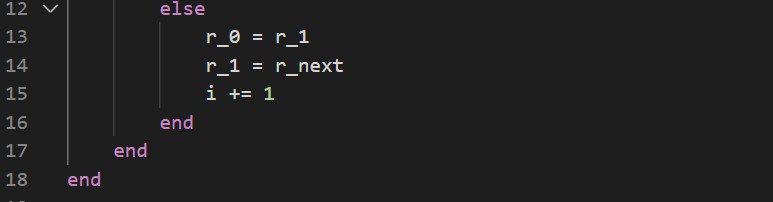


Механизм цикла: Этот цикл while true выполняется бесконечно долго, пока не завершится при обнаружении GCD. Вычисление остатка: r\_next присваивается значение r\_0 % r\_1, которое вычисляет остаток при делении r\_0 на r\_1.

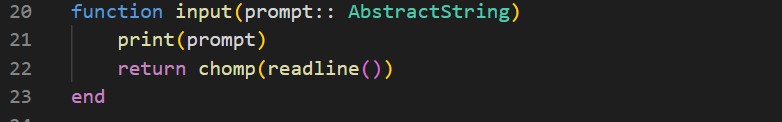
Условие завершения: Цикл останавливается, когда значение r\_next становится равным нулю, что означает, что значение r\_1 является GCD для исходных входных данных. Этот GCD возвращается в качестве результата.



Если значение r\_next не равно нулю, значение r\_0 изменяется на значение r\_1, а значение r\_1 изменяется на значение r\_next. В этой последовательности числа уменьшаются до тех пор, пока остаток не станет равным нулю. Результат: когда r\_next равно нулю, r\_1 является GCD и возвращается.

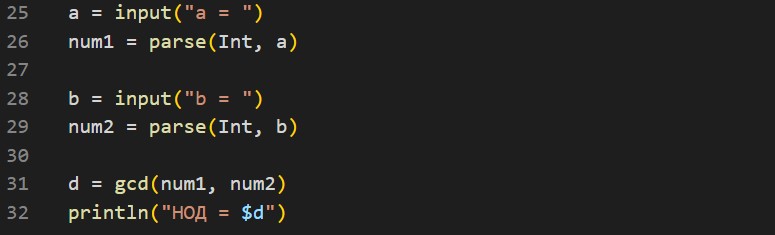


Функция ввода предлагает пользователю ввести значения, которые затем преобразуются в целые числа.



Пользователю предлагается ввести значение, которое хранится в a. Затем оно преобразуется в целое число num1 с помощью parse(Int, a). Аналогичным образом запрашивается другое значение, которое сохраняется в b и преобразуется в num2.

GCD чисел num1 и num2 вычисляется с использованием функции gcd.



Результат выводится на печать с сообщением, отображающим НОД, что на русском языке расшифровывается как GCD.

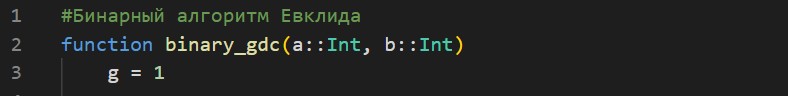


## Бинарный алгоритм Евклида

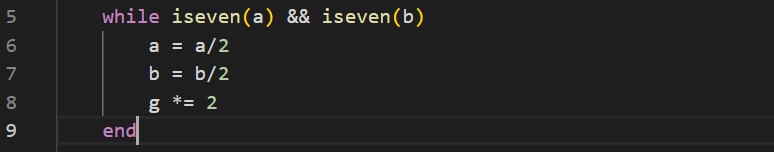
В этом отчете описывается реализация двоичного евклидова алгоритма (также известного как алгоритм

Штейна) в коде Julia для вычисления наибольшего общего делителя (НОД) двух целых чисел. Этот метод представляет собой вариацию традиционного евклидова алгоритма, использующего побитовые операции для более эффективной обработки четных и нечетных чисел.

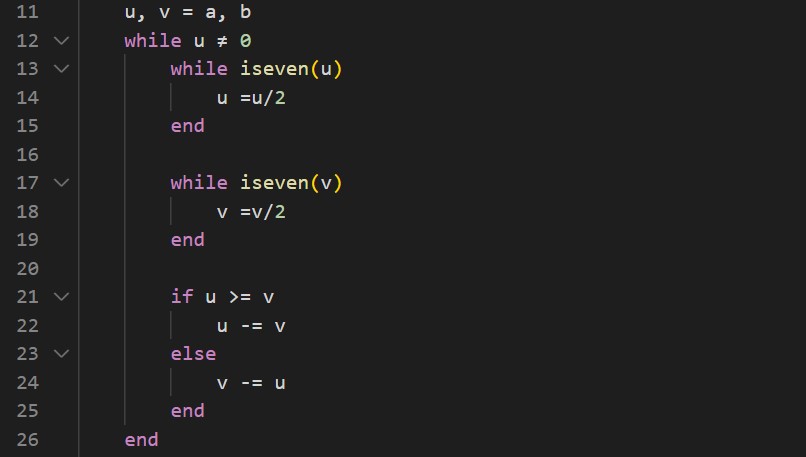
Функция binary\_gdc вычисляет GCD из двух целых чисел, используя двоичный алгоритм Евклида.



Хотя a и b четные, они делятся на 2, а g умножается на 2, чтобы отслеживать этот коэффициент. u and v are initialized to a and b, respectively, after their even factors have been divided out.Внешний цикл продолжается до тех пор, пока u не станет равным нулю. Делаем v нечетным: этот внутренний цикл продолжает делить v на 2 до тех пор, пока v не станет нечетным.



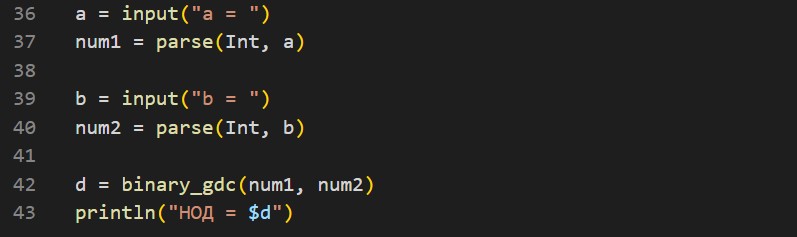
После того, как u и v станут нечетными, алгоритм использует вычитание для постепенного уменьшения большего значения: Если u больше или равно v, алгоритм вычитает v из u; в противном случае он вычитает u из v.



Как только u становится равным нулю, оставшееся значение v умножается на g (общий коэффициент 2s, удаленный ранее), чтобы получить окончательный GCD:



Программа запускается с запроса у пользователя двух целочисленных входных данных, Вызов binary\_gdc: Значение GCD для num1 и num2 вычисляется с помощью функции binary\_gdc.



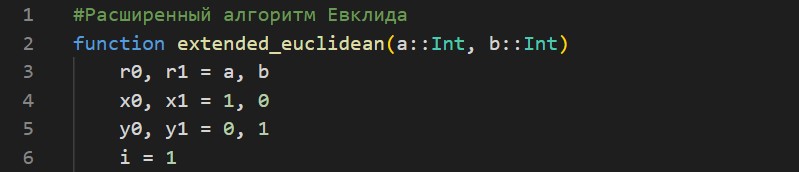
Результат, обозначенный как NOD (что в переводе на русский означает НОД), отображается пользователю.



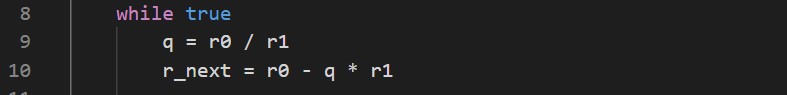
## Расширенный алгоритм Евклида

Функция extended\_euclidean является ядром этой программы. Она вычисляет как код двух целых чисел, так и коэффициенты для тождества Безу. Давайте рассмотрим каждую часть этой функции.

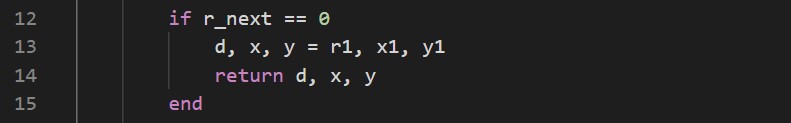
r0 и r1 хранят начальные значения a и b соответственно. x0 и x1 инициализируются как 1 и 0, соответственно, представляя начальные коэффициенты для a. y0 и y1 инициализируются как 0 и 1, представляя начальные коэффициенты для b. i - это счетчик итераций, который не является строго необходимым для работы алгоритма, но может быть полезен для отладки или отслеживания итераций.



На каждой итерации q вычисляется как целочисленное деление r0 на r1, представляющее, сколько раз r1 делится на r0. Следующий остаток r\_next вычисляется как r0 - q \* r1, что является ключевым шагом в евклидовом алгоритме для итеративного уменьшения значений до тех пор, пока остаток не достигнет нуля.

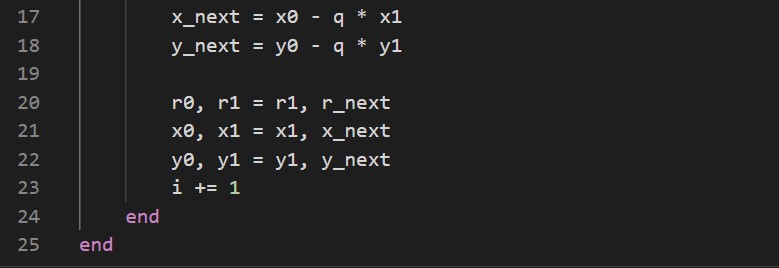


Когда r\_next равно нулю, это означает, что r1 содержит GCD из a и b. d присваивается значение r1, которое является GCD. x и y присваиваются значения x1 и y1, соответственно, представляющие собой конечные коэффициенты Безу.

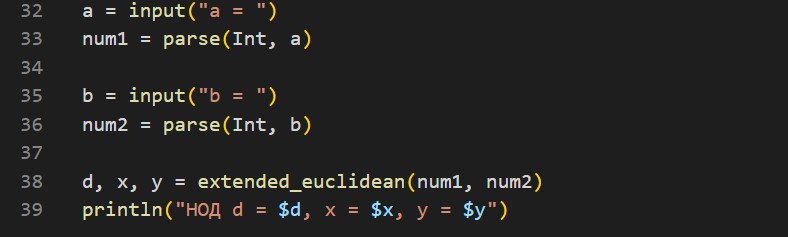


Следующие значения коэффициентов Безу вычисляются как x\_next = x0 - q \* x1 и y\_next = y0 - q \* y1. Эти значения выводятся из текущих коэффициентов и частного q, сохраняя связь x и y с исходными входными данными.

r0 и r1 обновляются значениями r1 и r\_next, постепенно приближаясь к нулевому остатку. Аналогично, x0 и x1 устанавливаются в значения x1 и x\_next, а y0 и y1 обновляются до y1 и y\_next. Счетчик итераций i увеличивается.



Программа предлагает пользователю ввести два значения, a и b, которые затем преобразуются в целые числа и присваиваются числам1 и num2 соответственно. Вызов extended\_euclidean: Функция вызывается с числами num1 и num2 в качестве аргументов, возвращая GCD d и коэффициенты x и y.

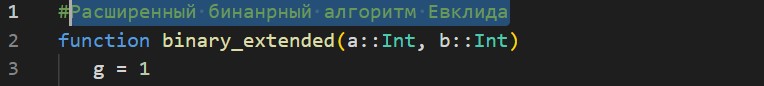


Инструкция println выводит результаты на русском языке с NOD d = (БОГ) и значениями x и y, показанными в качестве коэффициентов Безу.

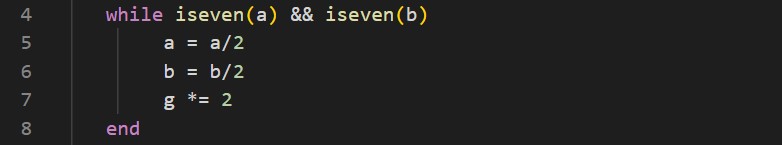


## Расширенный бинанрный алгоритм Евклида

Функция binary\_extended вычисляет коэффициенты GCD и Безу, используя расширенную версию алгоритма Штейна. Ниже приведена расшифровка функции:



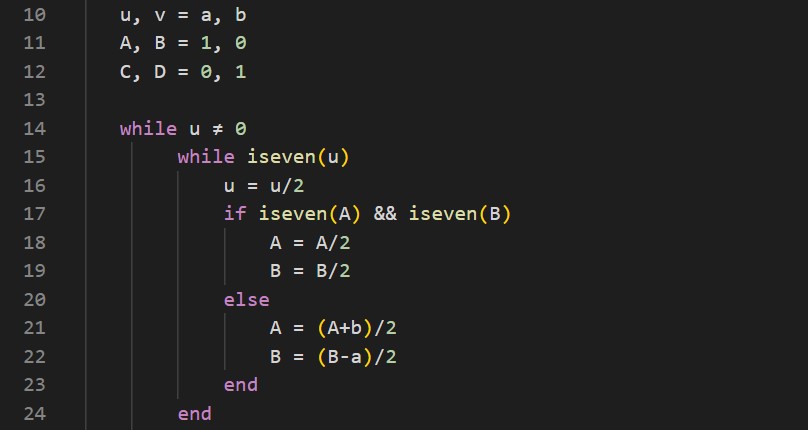
Удаление общих множителей, равных 2: Хотя a и b равны, они делятся на 2, а g умножается на 2, чтобы записать эти общие множители. Условие окончания цикла: Этот цикл завершается, как только значение a или b становится нечетным, удаляя все общие степени 2.



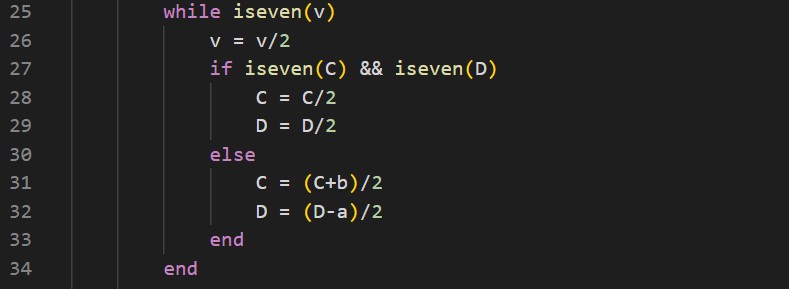
u и v инициализируются значениями a и b. A, B, C и D - коэффициенты для определения Безу.

Первоначально, А=1, B=0, C=0, и D=1, представляющий единичную матрицу.

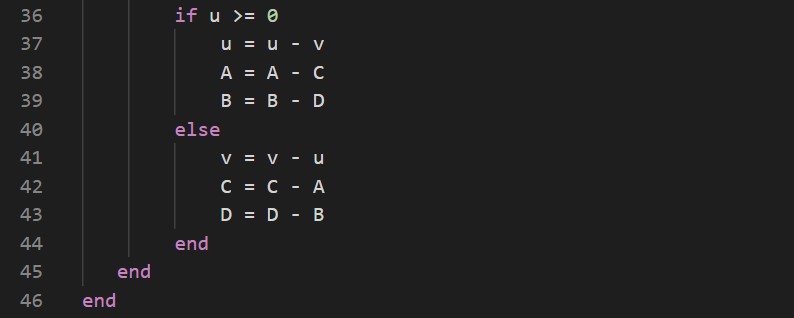
Внешний цикл продолжается до тех пор, пока u не станет равным нулю, что указывает на то, что НОД найден. Если A и B четные, они делятся на 2. В противном случае A и B корректируются для сохранения целочисленных значений путем добавления b к A и вычитания a из B, а затем деления обоих на 2. Аналогично, если C и D четные, они делятся на 2. В противном случае C и D корректируются таким образом, чтобы сохранить целочисленные значения, добавляя b к C и вычитая a из D.



Затем алгоритм сравнивает u и v, чтобы уменьшить их путем вычитания, а также обновляет коэффициенты. Если u больше или равно v, v вычитается из u, а коэффициенты A и B корректируются путем вычитания C и D. И наоборот, если v больше u, u вычитается из v, а C и D обновляются путем вычитания A и B.

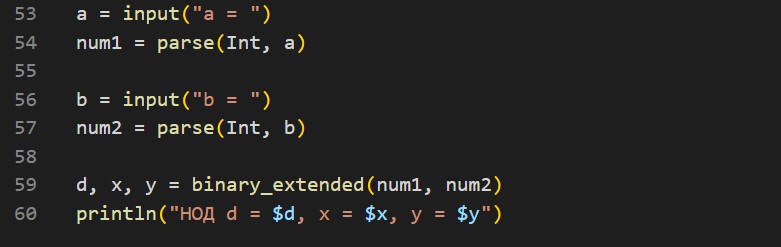


Вычисление конечного коэффициента полезного действия: Коэффициент полезного действия рассчитывается путем умножения v на g, которое включает все коэффициенты из 2, разделенные ранее.



Запрашивает у пользователя ввод данных: a и b извлекаются пользователем, сохраняются в виде строк и преобразуются в целые числа с помощью parse(Int, a) и parse(Int, b). Вызов функции binary\_extended:

Коэффициенты БОГА d и Безу x и y вычисляются с помощью функции binary\_extended.



# Выводы

Этот проект успешно демонстрирует реализацию и применение алгоритма Евклида и его расширенных версий, включая двоичный алгоритм Евклида и расширенный двоичный алгоритм Евклида. С помощью этих реализаций мы изучили различные эффективные методы вычисления наибольшего общего делителя (GOD) двух целых чисел, а также их коэффициентов Безу. Эти коэффициенты необходимы для выражения GCD в виде линейной комбинации исходных целых чисел, которая является фундаментальной при решении линейных диофантовых уравнений и имеет практическое применение в таких областях, как криптография и модульная арифметика.