Лабораторная работа №4

Математические основы защиты информации и информационной безопасности

Николаев Дмитрий Иванович, НПМмд-02-24

Содержание

# 1 Цель работы

Изучить работу алгоритмов вычисления наибольшего общего делителя: алгоритм Евклида, бинарный алгоритм Евклида, расширенный алгоритм Евклида, расширенный бинарный алгоритм Евклида, а также реализовать их программно.

# 2 Теоретическое введение

## 2.1 Вычисление наибольшего общего делителя

Пусть числа и целые и . Разделить на с остатком — значит представить 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, чисел , , равен . В качестве линейного представления можно взять:

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

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

## 2.2 Алгоритмы вычисления наибольшего общего делителя

Для вычисления наибольшего общего делителя двух целых чисел применяется способ повторного деления с остатком, называемый **алгоритмом Евклида**.

### 2.2.1 Алгоритм Евклида

**Вход:** целые числа , ; .

**Выход:** .

1. Положить , , .
2. Найти остаток от деления на .
3. Если , то положить . В противном случае положить и вернуться на шаг 2.
4. Результат: .

### 2.2.2 Бинарный алгоритм Евклида

**Бинарный алгоритм Евклида** является более быстрым при реализации на компьютере, поскольку использует двоичное представление чисел и . Он основан на следующих свойствах наибольшего общего делителя (считаем, что ):

1. Если оба числа и четные, то .
2. Если — нечетное, — четное, то .
3. Если оба числа и нечетные и , то .
4. Если , то .

**Вход:** целые числа , ; .

**Выход:** .

1. Положить .
2. Пока оба числа и четные, выполнять , , до получения хотя бы одного нечетного значения или .
3. Положить , .
4. Пока , выполнять следующие действия:
   1. Пока четное, полагать .
   2. Пока четное, полагать .
   3. При , положить . В противном случае положить .
5. Положить .
6. Результат: .

### 2.2.3 Расширенный алгоритм Евклида

**Вход:** целые числа , ; .

**Выход:** и такие целые числа , , что .

1. Положить , , , , , , .
2. Разделить с остатком на , получив и : .
3. Если , то положить , , . В противном случае положить , , и вернуться на шаг 2.
4. Результат: , , .

### 2.2.4 Расширенный бинарный алгоритм Евклида

**Вход:** целые числа , ; .

**Выход:** .

1. Положить .
2. Пока числа и четные, выполнять , , до получения хотя бы одного нечетного значения или .
3. Положить , , , , , .
4. Пока , выполнять следующие действия:
   1. Пока четное:
      1. Положить .
      2. Если и четные, то положить , . В противном случае положить , .
   2. Пока четное:
      1. Положить .
      2. Если и четные, то положить , . В противном случае положить , .
   3. Если , положить , , . В противном случае положить , , .
5. Положить , , .
6. Результат: , , .

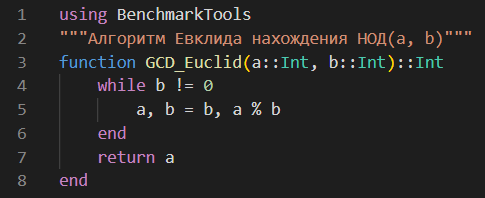
## 2.3 Резюме алгоритмов

1. **Алгоритм Евклида**: Это классический алгоритм, который повторяет деление с остатком, пока остаток не станет нулевым. Возвращает последний ненулевой остаток как НОД.
2. **Бинарный алгоритм Евклида**: Использует четность чисел и побитовые сдвиги для ускорения вычислений. Преимущество этого алгоритма заключается в эффективной работе на компьютерах с двоичной арифметикой.
3. **Расширенный алгоритм Евклида**: Помимо нахождения НОД, этот алгоритм вычисляет коэффициенты линейной комбинации .
4. **Расширенный бинарный алгоритм Евклида**: Сочетает подход бинарного алгоритма с расширенным, вычисляя также коэффициенты линейной комбинации, но с использованием более быстрых операций.

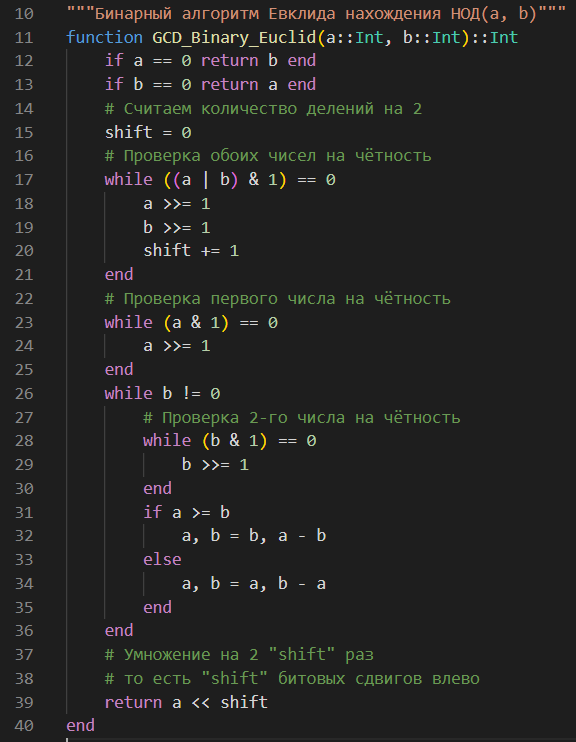
# 3 Выполнение лабораторной работы

Действуя согласно [1], реализуем все описанные алгоритмы на языке Julia.

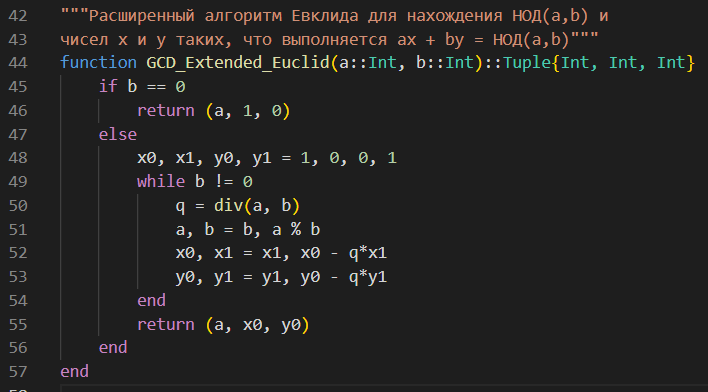
Программные реализации алгоритма Евклида (Рис.[??]), бинарного алгоритма Евклида (Рис.[??]), расширенного алгоритма Евклида (Рис.[??]) и расширенного бинарного алгоритма Евклида (Рис.[??,??]) представлены на соответствующих картинках. После чего на начальных данных , и с помощью пакета BenchmarkTools сравнены алгоритмы нахождения наибольшего общего делителя (Рис.[??]), где результаты представлены на Рис.[??].



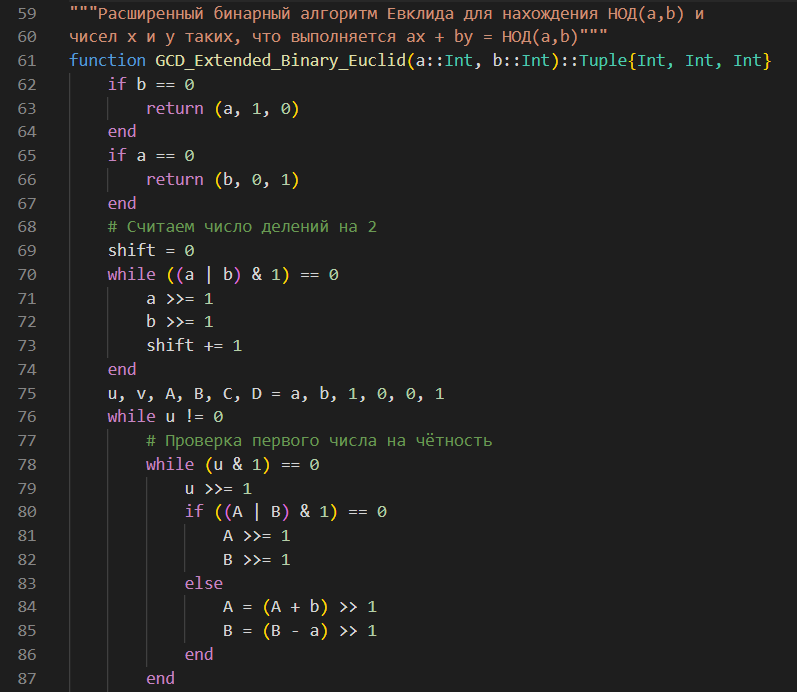
Код алгоритма Евклида на Julia



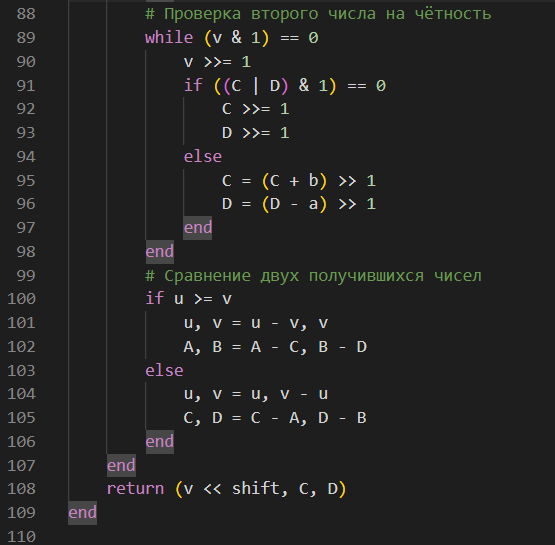
Код бинарного алгоритма Евклида на Julia



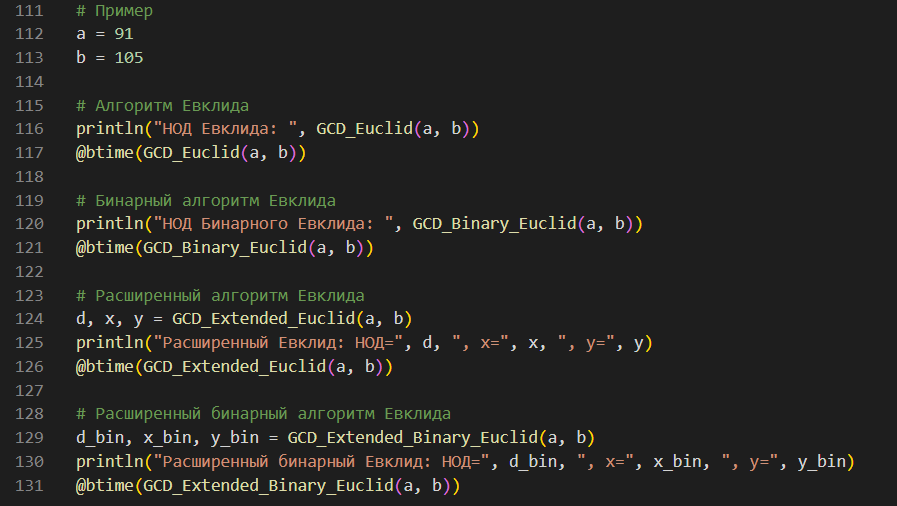
Код расширенного алгоритма Евклида на Julia



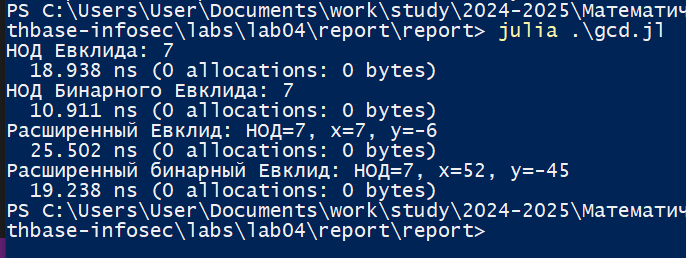
Код расширенного бинарного алгоритма Евклида на Julia (1/2)



Код расширенного бинарного алгоритма Евклида Евклида на Julia (2/2)



Начальные данные для сравнения алгоритмов нахождения НОД на Julia



Результат выполнения кода и сравнения алгоритмов нахождения НОД на Julia

# 4 Выводы

В ходе выполнения лабораторной работы я изучил работу алгоритмов вычисления наибольшего общего делителя: алгоритма Евклида, бинарного алгоритма Евклида, расширенного алгоритма Евклида, расширенного бинарного алгоритма Евклида, а также реализовал их программно.

# Список литературы

1. Лабораторная работа № 4. Вычисление наибольшего общего делителя [Электронный ресурс]. Саратовский государственный университет имени Н.Г. Чернышевского, 2024.