Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

**Лабораторная работа №1**

студента 5 курса 531 группы

специальности 10.05.01 «Компьютерная безопасность»

факультета компьютерных наук и информационных технологий

Енца Михаила Владимировича

|  |  |  |
| --- | --- | --- |
| Преподаватель  профессор | \_\_\_\_\_\_\_\_\_\_ | В. А.  Молчанов |
|  | подпись, дата |  |
| Заведующий кафедрой  д.ф.-м.н., доцент | \_\_\_\_\_\_\_\_\_\_ | М. Б. Абросимов |
|  | подпись, дата |  |

Саратов 2018

**1. Постановка задачи.**

Изучение основных операции над целыми числами и их программная реализация.

Порядок выполнения работы: разобрать алгоритмы Евклида (обычный, бинарный и расширенный) вычисления НОД целых чисел и привести их программную реализацию; разобрать алгоритмы решения систем сравнений (с помощью греко-китайской теоремы и методом Гарнера).

**2. Теоретические сведения по рассмотренным темам с их обоснованием.**

**Определение**. Целое число называется наибольшим общим делителем чисел (обозначается ), если выполняются следующие условия:

1) каждое из чисел делится на ;

2) если – другой общий делитель чисел , то делится на .

**Теорема (о единственности наибольшего общего делителя)**

Пусть числа целые и – их наибольший общий делитель. Целое число является наибольшим общим делителем чисел тогда и только тогда, когда .

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

При условии попарной взаимной простоты чисел для любых существует единственное решение системы уравнений

**3. Результаты работы.**

**Описание алгоритмов Евклида вычисления НОД целых чисел.**

*Обычный алгоритм Евклида*: для вычисления наибольшего общего делителя двух целых чисел применяется способ повторного деления с остатком, называемый алгоритмом Евклида.

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

1) Если оба числа и четные, то ;

2) Если число нечетное, число четное, то ;

3) Если оба числа a и b нечетные, , то ;

4) Если , то .

*Расширенный алгоритм Евклида*: алгоритм находит наибольший общий делитель чисел и и его линейное представление, то есть целые числа и , для которых .

**Описание алгоритмов решения систем сравнений**

Т. Китайская теорема об остатках.

Пусть , числа попарно взаимно просты, и , . Тогда решение системы сравнений (1) находится по формуле

Алгоритм Гарнера: Пусть , числа попарно взаимно просты, и .

Тогда решение системы (1) может быть представлено в виде

Где , и числа вычисляются по формулам

**Псевдокоды рассмотренных алгоритмов**

Псевдокод обычного алгоритма Евклида.

.

Псевдокод бинарного алгоритма Евклида.

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.

4.1. Пока четное, полагать .

4.2. Пока четное, полгать .

4.3. При положить . В противном случае положить .

5. Положить .

6. Результат: .

Псевдокод расширенного алгоритма Евклида.

Вход. Целые числа .

Выход. ; такие целые числа , что .

1. Положить .

2. Разделить с остатком на : .

3. Если , то положить . В противном случае положить и вернуться на шаг 2.

4. Результат: .

Псевдокод китайской теоремы об остатках.

Вход. Массивы ;

Выход. Решение системы сравнений (1).

1. ,

2.

3. Для всех выполнять шаг 4

4.

5. Результат:

Псевдокод алгоритма Гарнера.

Вход. Массивы ;

Выход. Решение системы сравнений (1).

1. Для всех выполнять шаги 2-4

2.

3. Для всех выполнять шаг 4

4. ,

5.

6. Для всех выполнять шаг 7

7.

8.

9. Результат:

**Коды программ, реализующей рассмотренные алгоритмы**

Программа реализована на языке Python (версия интерпретатора 3.6).

Файл main.py

import functools

from operator import mul

import utils

def euclid(a, b):

assert 0 < b <= a

r = [a, b]

i = 1

while True:

r.append(r[i - 1] % r[i])

if r[i + 1] == 0:

d = r[i]

break

i += 1

return d

def euclid\_bin(a, b):

assert 0 < b <= a

g = 1

while a % 2 == 0 and b % 2 == 0:

a >>= 1

b >>= 1

g <<= 1

u = a

v = b

while u != 0:

while u % 2 == 0:

u >>= 1

while v % 2 == 0:

v >>= 1

if u >= v:

u = u - v

else:

v = v - u

d = g \* v

return d

def euclid\_extended(a, b):

assert 0 < b <= a

r = [a, b]

x = [1, 0]

y = [0, 1]

i = 1

q = [0]

while True:

q.append(r[i - 1] // r[i])

r.append(r[i - 1] % r[i])

if r[i + 1] == 0:

d = r[i]

X = x[i]

Y = y[i]

break

else:

x.append(x[i - 1] - q[i] \* x[i])

y.append(y[i - 1] - q[i] \* y[i])

i += 1

return d, X, Y

def chinese\_theorem(a, e):

m = functools.reduce(mul, e)

u = 0

for idx in range(len(a)):

u += a[idx] \* (m // e[idx]) \* utils.inverse(m // e[idx], e[idx])

return u % m

def garner(r, m):

n = len(m)

c = [0] \* n

for i in range(1, n):

c[i] = 1

for j in range(0, i):

u = utils.inverse(m[j], m[i])

c[i] = u \* c[i] % m[i]

u = r[0]

x = u

for i in range(1, n):

u = (r[i] - x) \* c[i] % m[i]

x = x + u \* functools.reduce(mul, (m[idx] for idx in range(i)), 1)

return x

def main():

"""

simple for Garner and Chine theorem

a: 2 1 3 8

m: 5 7 11 13

"""

mod = int(input('1 - Chine theorem, 2 - Garner, 3 - Euclides '))

if mod != 3:

a = [int(item) for item in input('a: ').split()]

m = [int(item) for item in input('m: ').split()]

if mod == 1:

print(chinese\_theorem(a, m))

if mod == 2:

print(garner(a, m))

else:

a, b = list(map(int, input('a, b: ').split()))

print('gcd: ', euclid(a, b))

print('gcd: ', euclid\_bin(a, b))

print('gcd, x, y: ', euclid\_extended(a, b))

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Оценки сложности рассмотренных алгоритмов**

Сложность обычного алгоритма Евклида: .

Сложность бинарного алгоритма Евклида: .

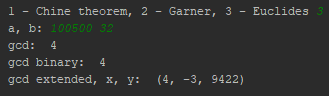
Сложность расширенного алгоритма Евклида: .

Сложность китайской теоремы об остатках:

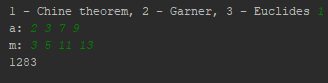
Сложность алгоритма Гарнера:

**Результаты тестирования программ**

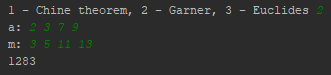
Алгоритмы Евклида:



Исполнение алгоритма на основе китайской теоремы об остатках:



Исполнение алгоритма Гарнера:



Пример работы алгоритмов Евклида

|  |  |  |
| --- | --- | --- |
|  | Числа | |
| Алгоритм Евклида | 35, 14 | 232564, 7679 |
| Обычный | 7 | 1097 |
| Бинарный | 7 | 1097 |
| Расширенный |  | 1097, -3, 91 |

Оценка скорости алгоритмов производилась с использованием библиотечной функции языка Python для измерения времени совместно с различными искусственными способами замедления работы программы. Первый из них (тест 1) – многократное повторение (1000000 проходов) вызова функции вычисления наибольшего общего делителя для одних и тех же входных данных. Тесты такого рода позволяют экспериментально установить зависимость скорости работы алгоритмов от вида входных данных. Второй (тест 2) – вычисление наибольшего общего делителя всех комбинаций чисел из достаточно большого диапазона. Такой тест позволит получить общие характеристики алгоритмов.

Результаты тестирования алгоритмов на быстродействие

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Тест 1 | | Тест 2 | |
|  | Числа | | Диапазон | |
| Алгоритм Евклида | 321, 123 | 16777217, 1023 | 1:1000 | 20000:21000 |
| Обычный | 7.18 | 5.44 | 3.09 | 3.57 |
| Бинарный | 8.15 | 31.58 | 5.74 | 8.76 |
| Расширенный | 19.74 | 15.83 | 8.19 | 10.25 |