Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №4**

**Дисциплина: Криптографические протоколы**

Работу выполнила: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д. Н. Баева

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А. А. Крамаренко

**Цель работы:** выделить циклотомические классы и найти соответствующие им минимальные многочлены для поля GF(16) для образующего многочлена 11001.

**Ход работы:**

Реализация задачи состоит из применения следующих функций:

1. Функция gcd(a, b) реализует алгоритм нахождения наибольшего общего делителя (НОД) двух чисел. Она использует цикл while, чтобы последовательно вычислять остаток от деления a на b. Когда остаток становится равным нулю, функция возвращает a — это и есть НОД.

2. Функция find\_primitive\_elements() находит примитивные элементы в поле GF(16). Примитивные элементы в конечном поле (поле Галуа) — это такие элементы, которые образуют циклическую группу порядка, равного размеру поля минус 1. Она использует цикл для проверки каждого числа от 1 до 15 и добавляет примитивные элементы в список. Функция представлена на рисунке 1.

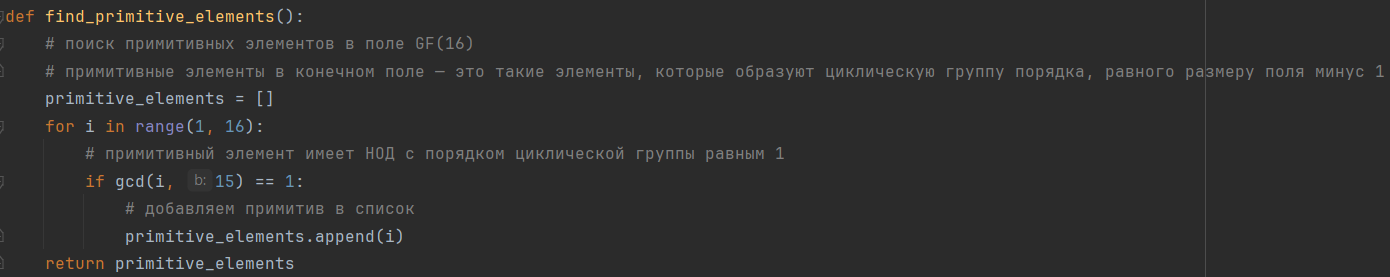


Рисунок 1 – Поиск примитивных элементов в поле.

3. Функция find\_cyclotomic\_classes(primitive\_elements) находит циклотомические классы – рисунок 2. Для каждого примитивного элемента она строит циклотомический класс, добавляя элементы, полученные умножением на самого себя. Цикл продолжается, пока не получится 1. Все циклотомические классы сохраняются в списке.

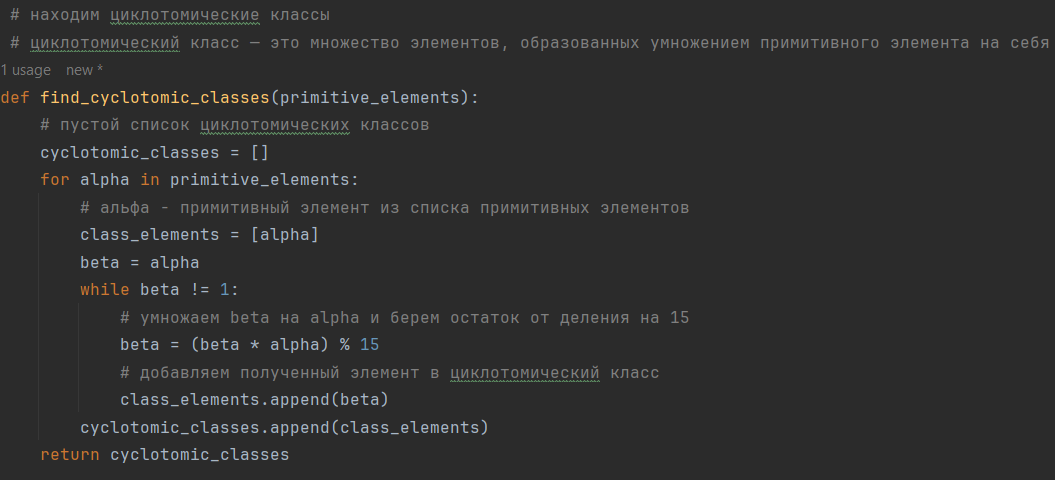


Рисунок 2 – Поиск цикломатических классов.

4. Функция find\_minimal\_polynomials(cyclotomic\_classes) находит минимальные многочлены для каждого циклотомического класса – рисунок 3. Для каждого класса она создает список коэффициентов, где индекс элемента соответствует значению элемента в классе. Затем она убирает лишние степени (не более 4) и сохраняет коэффициенты в список минимальных многочленов.

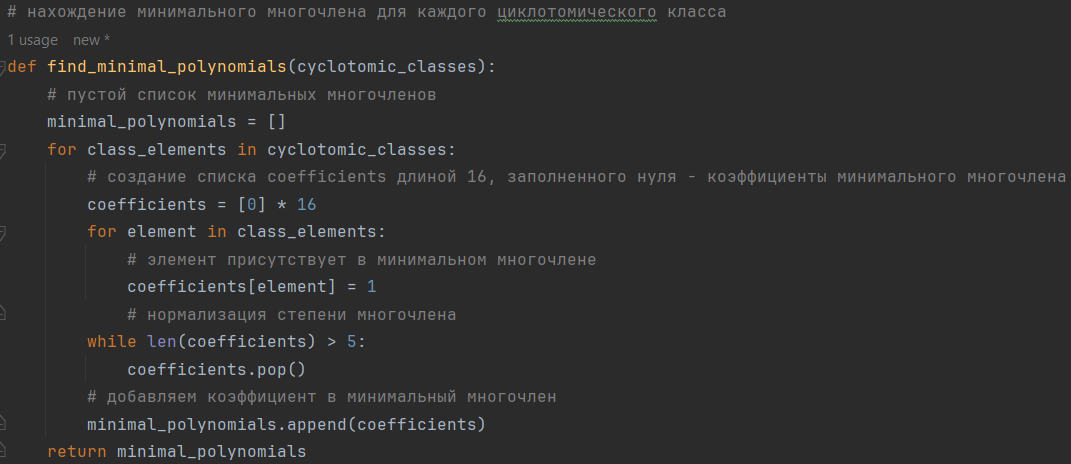


Рисунок 3 – Поиск минимального многочлена.

В финале программы следует методы, запрашивающие у пользователя коэффициенты образующего многочлена. Затем названные выше функции вызываются и выводится результаты на экран, а также выполняется проверка, что минимальные многочлены делятся на образующий многочлен (рисунок 4).

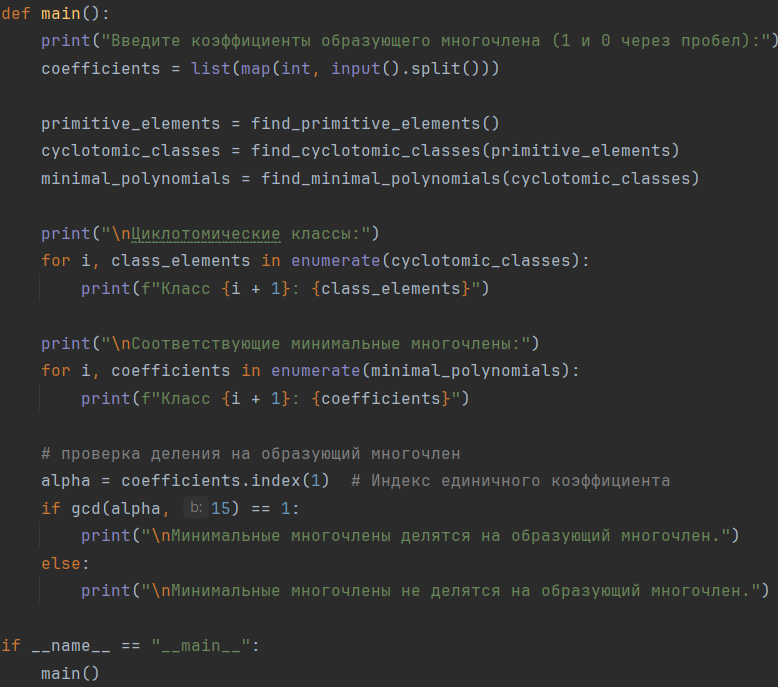


Рисунок 4 – Main-функция.

**Листинг программы**

Файл LW\_4.py

# # выделить циклотомические классы

# # найти соответствующие им минимальные многочлены

# # для поля GF(16) для образующего многочлена 11001

def gcd(a, b):

while b:

a, b = b, a % b

return a

def find\_primitive\_elements():

# поиск примитивных элементов в поле GF(16)

# примитивные элементы в конечном поле — это такие элементы, которые образуют циклическую группу порядка, равного размеру поля минус 1

primitive\_elements = []

for i in range(1, 16):

# примитивный элемент имеет НОД с порядком циклической группы равным 1

if gcd(i, 15) == 1:

# добавляем примитив в список

primitive\_elements.append(i)

return primitive\_elements

# находим циклотомические классы

# циклотомический класс — это множество элементов, образованных умножением примитивного элемента на себя

def find\_cyclotomic\_classes(primitive\_elements):

# пустой список циклотомических классов

cyclotomic\_classes = []

for alpha in primitive\_elements:

# альфа - примитивный элемент из списка примитивных элементов

class\_elements = [alpha]

beta = alpha

while beta != 1:

# умножаем beta на alpha и берем остаток от деления на 15

beta = (beta \* alpha) % 15

# добавляем полученный элемент в циклотомический класс

class\_elements.append(beta)

cyclotomic\_classes.append(class\_elements)

return cyclotomic\_classes

# нахождение минимального многочлена для каждого циклотомического класса

def find\_minimal\_polynomials(cyclotomic\_classes):

# пустой список минимальных многочленов

minimal\_polynomials = []

for class\_elements in cyclotomic\_classes:

# создание списка coefficients длиной 16, заполненного нуля - коэффициенты минимального многочлена

coefficients = [0] \* 16

for element in class\_elements:

# элемент присутствует в минимальном многочлене

coefficients[element] = 1

# нормализация степени многочлена

while len(coefficients) > 5:

coefficients.pop()

# добавляем коэффициент в минимальный многочлен

minimal\_polynomials.append(coefficients)

return minimal\_polynomials

def main():

print("Введите коэффициенты образующего многочлена (1 и 0 через пробел):")

coefficients = list(map(int, input().split()))

primitive\_elements = find\_primitive\_elements()

cyclotomic\_classes = find\_cyclotomic\_classes(primitive\_elements)

minimal\_polynomials = find\_minimal\_polynomials(cyclotomic\_classes)

print("\nЦиклотомические классы:")

for i, class\_elements in enumerate(cyclotomic\_classes):

print(f"Класс {i + 1}: {class\_elements}")

print("\nСоответствующие минимальные многочлены:")

for i, coefficients in enumerate(minimal\_polynomials):

print(f"Класс {i + 1}: {coefficients}")

# проверка деления на образующий многочлен

alpha = coefficients.index(1) # Индекс единичного коэффициента

if gcd(alpha, 15) == 1:

print("\nМинимальные многочлены делятся на образующий многочлен.")

else:

print("\nМинимальные многочлены не делятся на образующий многочлен.")

if \_\_name\_\_ == "\_\_main\_\_":

main()