МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КУБГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**Отчет**

**по практическому заданию №4**

**по курсу**

**«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»**

Работу выполнил

Студенты 46 группы

Нагалевский А.М.

Преподаватель:

Крамаренко А.А.

**Цель работы:** реализовать программный продукт для решения задач над некоторым полем Галуа для ускорения решения задач из лабораторной работы.

**Ход работы:**

№ 1. Выделить циклотомические классы и найти соответствующие им минимальные многочлены для поля GF(16) для образующего многочлена 11001

Для решения данной задачи выполним следующие шаги:

* Определение циклотомических классов:
* Циклотомические классы определяются в поле GF(2n) для n-степенного многочлена. В данном случае, для GF(16), n=4, так как 16=2^4.
* Циклотомические классы определяются по модулю 2^n - 1, в нашем случае, по модулю 15.
* Для каждого элемента в {0, 1, ..., 14} вычислим его множество, возводя его в степени 2^k до тех пор, пока не получим начальное число, все операции выполняются по модулю 15.

Цикломатические классы для GF(2^4):



Чтобы показать все 16 элементов поля GF(16), используя образующий многочлен (в двоичной форме 11001), мы можем начать с определения элемента , который является корнем данного образующего многочлена, и затем генерировать все элементы поля, последовательно умножая на себя и редуцируя результаты с помощью образующего многочлена при необходимости.

В поле GF(16), элементы можно представить как многочлены степени меньше 4 над GF(2). Таким образом, элементы поля могут быть записаны как , где , что следует из образующего многочлена.

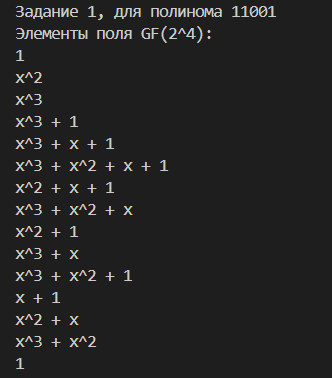
Для сложения используется XOR (исключающее ИЛИ), так как операция сложения в GF(2^n) эквивалентна сложению без переноса.

Вот как мы можем генерировать и показывать все элементы поля GF(16):

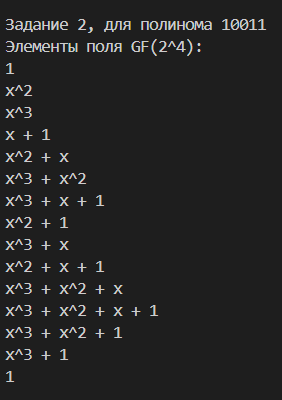
Начнем с 1 (единица в двоичном виде: 0001) и (которое мы можем считать равным 0010, представляющее ).

Последовательно умножаем на себя, редуцируя результат через образующий многочлен при необходимости, чтобы поддерживать степень многочлена меньше 4.

Применяем операцию XOR для имитации сложения многочленов по модулю 2, чтобы получить следующий элемент поля при каждом умножении.



№ 2. Выделить циклотомические классы и найти соответствующие им минимальные многочлены для поля GF(16) для образующего многочлена 10011



№ 3-7. Для многочлена над полем с образующим многочленом найти циклотомический класс и минимальный многочлен.

Задание 3-6

Цикломатические классы для GF(2^5):

[[1, 2, 4, 8, 16], [3, 6, 12, 16, 24], [5, 8, 10, 16, 20], [7, 14, 16, 24, 28], [4, 8, 9, 16, 18], [11, 12, 16, 22, 24], [8, 13, 16, 20, 26], [15, 16, 24, 28, 30], [2, 4, 8, 16, 17], [6, 12, 16, 19, 24], [8, 10, 16, 20, 21], [14, 16, 23, 24, 28], [4, 8, 16, 18, 25], [12, 16, 22, 24, 27], [8, 16, 20, 26, 29], [16, 24, 28, 30, 31]]

Циклотомический класс и минимальный многочлен для x:

Элементы поля GF(2^5):

1

x^2

x^3

x^4

x^2 + 1

x^3 + x

x^4 + x^2

x^3 + x^2 + 1

x^4 + x^3 + x

x^4 + 1

x^2 + x + 1

x^3 + x^2 + x

x^4 + x^3 + x^2

x^4 + x^3 + x^2 + 1

x^4 + x^3 + x^2 + x + 1

x^4 + x^3 + x + 1

x^4 + x + 1

x + 1

x^2 + x

x^3 + x^2

x^4 + x^3

x^4 + x^2 + 1

x^3 + x^2 + x + 1

x^4 + x^3 + x^2 + x

x^4 + x^3 + 1

x^4 + x^2 + x + 1

x^3 + x + 1

x^4 + x^2 + x

x^3 + 1

x^4 + x

1

Циклотомический класс и минимальный многочлен для x^3:

Элементы поля GF(2^5):

1

x^3

x^4

x^2 + 1

x^3 + x

x^4 + x^2

x^3 + x^2 + 1

x^4 + x^3 + x

x^4 + 1

x^2 + x + 1

x^3 + x^2 + x

x^4 + x^3 + x^2

x^4 + x^3 + x^2 + 1

x^4 + x^3 + x^2 + x + 1

x^4 + x^3 + x + 1

x^4 + x + 1

x + 1

x^2 + x

x^3 + x^2

x^4 + x^3

x^4 + x^2 + 1

x^3 + x^2 + x + 1

x^4 + x^3 + x^2 + x

x^4 + x^3 + 1

x^4 + x^2 + x + 1

x^3 + x + 1

x^4 + x^2 + x

x^3 + 1

x^4 + x

1

x

Циклотомический класс и минимальный многочлен для x^5:

Элементы поля GF(2^5):

1

x^2 + 1

x^3 + x

x^4 + x^2

x^3 + x^2 + 1

x^4 + x^3 + x

x^4 + 1

x^2 + x + 1

x^3 + x^2 + x

x^4 + x^3 + x^2

x^4 + x^3 + x^2 + 1

x^4 + x^3 + x^2 + x + 1

x^4 + x^3 + x + 1

x^4 + x + 1

x + 1

x^2 + x

x^3 + x^2

x^4 + x^3

x^4 + x^2 + 1

x^3 + x^2 + x + 1

x^4 + x^3 + x^2 + x

x^4 + x^3 + 1

x^4 + x^2 + x + 1

x^3 + x + 1

x^4 + x^2 + x

x^3 + 1

x^4 + x

1

x

x^2

x^3

Циклотомический класс и минимальный многочлен для x^7:

Элементы поля GF(2^5):

1

x^4 + x^2

x^3 + x^2 + 1

x^4 + x^3 + x

x^4 + 1

x^2 + x + 1

x^3 + x^2 + x

x^4 + x^3 + x^2

x^4 + x^3 + x^2 + 1

x^4 + x^3 + x^2 + x + 1

x^4 + x^3 + x + 1

x^4 + x + 1

x + 1

x^2 + x

x^3 + x^2

x^4 + x^3

x^4 + x^2 + 1

x^3 + x^2 + x + 1

x^4 + x^3 + x^2 + x

x^4 + x^3 + 1

x^4 + x^2 + x + 1

x^3 + x + 1

x^4 + x^2 + x

x^3 + 1

x^4 + x

1

x

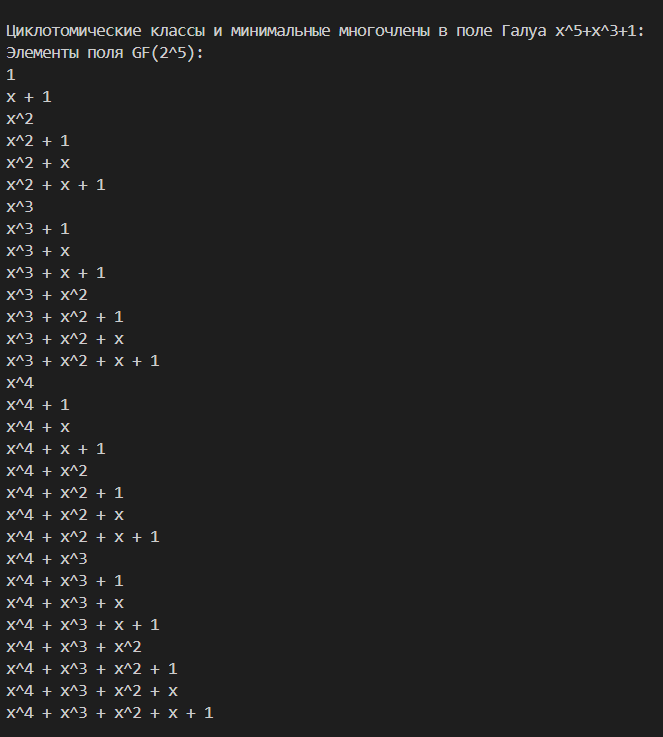
x^2

x^3

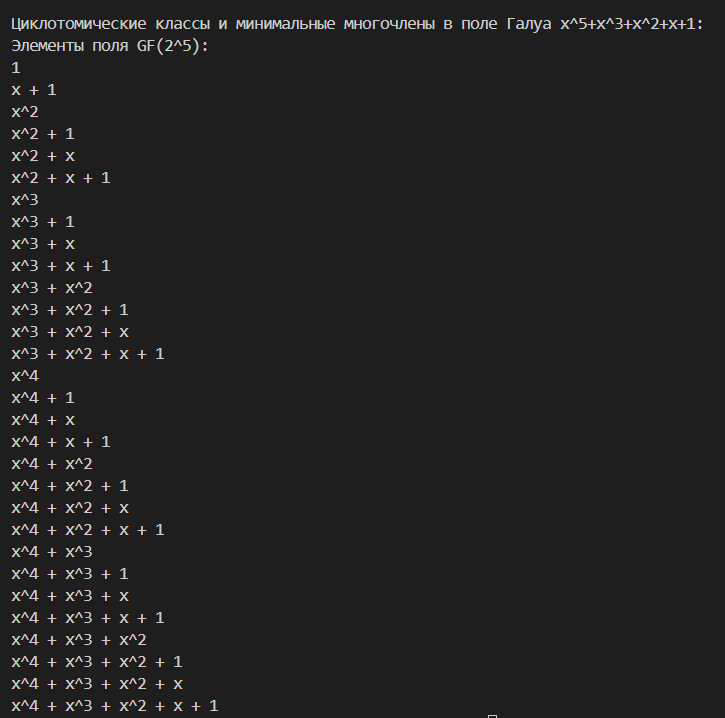
x^4

x^2 + 1

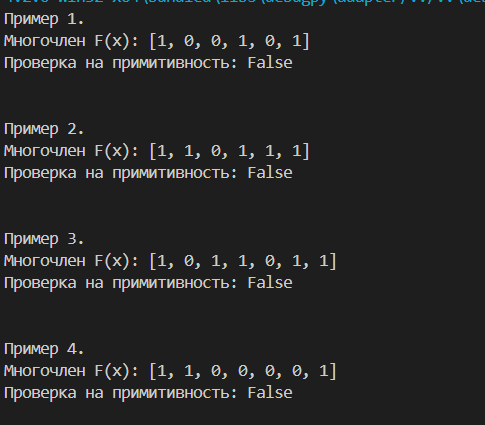
№ 7 Построить циклотомические классы и минимальные многочлены в поле Галуа.



№ 8 Построить циклотомические классы и минимальные многочлены в поле Галуа.



№ 9-12 Проверить на примитивность неприводимый многочлен 100101, 110111, 1011011, 1100001



**Листинг программы**

Файл task1-8.py:

def find\_cyclotomic\_classes(n):

    modulus = 2 \*\* n - 1

    classes = []

    found = set()

    for i in range(1, modulus + 1):  # Учитываем все элементы до 2^n - 1

        if i not in found:

            current\_class = set()

            value = i

            while value not in current\_class:

                current\_class.add(value)

                found.add(value)

                value = (value \* 2) % (modulus + 1)

            classes.append(sorted(current\_class)[1:])

    print(f"Цикломатические классы для GF(2^{n}):")

    return classes

def binary\_to\_polynomial(binary):

    binary\_str = bin(binary)[2:]

    result = ""

    degree = len(binary\_str) - 1

    for i, bit in enumerate(binary\_str):

        if bit == '1':

            if degree - i > 1:

                result += f"x^{degree - i} + "

            elif degree - i == 1:

                result += "x + "

            else:

                result += "1"

    if result.endswith(" + "):

        result = result[:-3]

    return result

def multiply\_by\_alpha(alpha, modulus, n):

    # Умножение alpha на x с учетом степени n для GF(2^n)

    result = alpha << 1

    # Проверяем, превышает ли результат предел для GF(2^n)

    if result >= 2 \*\* n:

        result ^= modulus  # Редукция по модулю образующего многочлена (XOR)

    return result

def reduce\_alpha(alpha, modulus, m):

    max\_degree = 2\*\*(m)  # Максимально возможное значение для GF(2^m) равно 2^m, а не 2^(m-1)

    while alpha >= max\_degree:

        # Находим степень текущего alpha

        degree\_alpha = alpha.bit\_length() - 1

        # Находим степень модуля

        degree\_modulus = modulus.bit\_length() - 1

        # Проверяем, что degree\_alpha >= degree\_modulus для корректного сдвига

        if degree\_alpha >= degree\_modulus:

            # Сдвигаем модуль на разницу степеней

            shifted\_modulus = modulus << (degree\_alpha - degree\_modulus)

            # Применяем XOR для редукции

            alpha ^= shifted\_modulus

        else:

            # Если alpha меньше степени модуля, дальнейшая редукция не требуется

            break

    return alpha

def generate\_elements\_and\_minimal\_polynomials(n, modulus, alpha=0b10):

    # Начинаем с 1 (единичный многочлен)

    elements = [1]

    alpha = reduce\_alpha(alpha, modulus, n)

    # Начальное значение alpha по умолчанию: (x)

    for \_ in range(1, 2 \*\* n - 1):  # Генерируем все элементы поля GF(2^n)

        alpha = multiply\_by\_alpha(alpha, modulus, n)

        elements.append(alpha)

    # Выводим элементы поля и их полиномиальное представление

    print("Элементы поля GF(2^{}):".format(n))

    for element in elements:

        print(binary\_to\_polynomial(element))

#task 1-2

print(find\_cyclotomic\_classes(4))

print("Задание 1, для полинома 11001")

generate\_elements\_and\_minimal\_polynomials(4,0b11001)

print("\nЗадание 2, для полинома 10011")

generate\_elements\_and\_minimal\_polynomials(4,0b10011)

#task 3-6

print("\nЗадание 3-6")

print(find\_cyclotomic\_classes(5))

print("Циклотомический класс и минимальный многочлен для x:")

generate\_elements\_and\_minimal\_polynomials(5,0b100101,0b10)

print("\nЦиклотомический класс и минимальный многочлен для x^3:")

generate\_elements\_and\_minimal\_polynomials(5,0b100101,0b100)

print("\nЦиклотомический класс и минимальный многочлен для x^5:")

generate\_elements\_and\_minimal\_polynomials(5,0b100101,0b10000)

print("\nЦиклотомический класс и минимальный многочлен для x^7:")

generate\_elements\_and\_minimal\_polynomials(5,0b100101,0b1000000)

#task 7-8

def generate\_elements\_and\_minimal\_polynomials\_7(m, modulus, alpha\_start):

    # переберем все alpha и найдем все уникальные элементы поля

    # Начинаем с начального значения alpha

    alpha = alpha\_start

    elements = [1]  # 1 всегда присутствует в поле GF(2^m)

    # Генерируем элементы поля GF(2^m)

    for \_ in range(1, 2\*\*m - 1):

        alpha = reduce\_alpha(multiply\_by\_alpha(alpha, modulus, m), modulus, m)

        if alpha not in elements:  # Убедимся, что элементы уникальны

            elements.append(alpha)

    # Выводим элементы поля и их полиномиальное представление

    print("Элементы поля GF(2^{}):".format(m))

    elements.sort()

    for element in elements:

        print(binary\_to\_polynomial(element))

print("\nЗадание 7")

print("\nЦиклотомические классы и минимальные многочлены в поле Галуа x^5+x^3+1: ")

generate\_elements\_and\_minimal\_polynomials\_7(5, 0b101001, 0b10)  # для alpha = x

print("\nЗадание 8")

print("\nЦиклотомические классы и минимальные многочлены в поле Галуа x^5+x^3+x^2+x+1: ")

generate\_elements\_and\_minimal\_polynomials\_7(5, 0b101111, 0b10)  # для alpha = x

Файл task8-12.py:

# Примеры из задачи

is\_irreducible\_examples = [

    [1, 0, 0, 1, 0, 1],

    [1, 1, 0, 1, 1, 1],

    [1, 0, 1, 1, 0, 1, 1],

    [1, 1, 0, 0, 0, 0, 1],

]

irr\_polynomials = [

    [1, 1, 1],  # x^2 + x + 1

    [1, 0, 1, 1],  # x^3 + x + 1

    [1, 0, 0, 1, 1],  # x^4 + x + 1

    [1, 0, 0, 1, 0, 1],  # x^5 + x^2 + 1

    [1, 0, 0, 0, 0, 1, 1],  # x^6 + x + 1

    [1, 0, 0, 0, 0, 0, 1, 1],  # x^7 + x + 1

    [1, 0, 0, 0, 1, 1, 1, 0, 1],  # x^8 + x^4 + x^3 + x^2 + 1

    [1, 0, 0, 0, 0, 1, 0, 0, 0, 1],  # x^9 + x^4 + 1

    [1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1],  # x^10 + x^3 + 1

    [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1],  # x^11 + x^2 + 1

    [1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1],  # x^12 + x^6 + x^4 + x + 1

    [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1],  # x^13 + x^4 + x^3 + x + 1

    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1],  # x^14 + x^10 + x^ 6 + x + 1

    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1],  # x^15 + x + 1

    [1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1],  # x^16 + x^12 + x^3 + x + 1

]

def is\_irreduciblle(poly):

    # Полиномы степени 0 и 1 всегда неприводимы

    if len(poly) <= 2:

        return True

    elif len(poly) > 16:

        print("Максимальная степень до степени 16.")

        return False

    # Проверяем содержит ли irr\_polynomials полином

    if poly in irr\_polynomials:

        return True

    else:

        return False

def prime\_factors(n):

    i = 2

    factors = []

    while i \* i <= n:

        if n % i:

            i += 1

        else:

            n //= i

            factors.append(i)

    if n > 1:

        factors.append(n)

    return factors

def gf2\_poly\_powmod(x, k, mod\_poly):

    result = [1]  # многочлен степени 0

    while k > 0:

        if k & 1:

            result = gf2\_poly\_mod(gf2\_poly\_mul(result, x), mod\_poly)

        x = gf2\_poly\_mod(gf2\_poly\_mul(x, x), mod\_poly)

        k >>= 1

    return result

def gf2\_poly\_mod(poly, mod\_poly):

    def degree(p):

        while p and p[-1] == 0:

            p.pop()  # Удаляем нулевые коэффициенты с конца

        return len(p) - 1

    dp = degree(poly)

    dm = degree(mod\_poly)

    while dp >= dm:

        diff = [0]\*(dp - dm) + mod\_poly

        for i in range(len(poly)):

            poly[i] ^= diff[i]

        dp = degree(poly)

    return poly

def gf2\_poly\_mul(a, b):

    result = [0] \* (len(a) + len(b) - 1)

    for i, coeff\_a in enumerate(a):

        for j, coeff\_b in enumerate(b):

            result[i + j] ^= (coeff\_a & coeff\_b)

    return result

def is\_primitive(poly):

    if not is\_irreduciblle(poly):

        return False

    n = len(poly) - 1  # Степень многочлена

    order = 2 \*\* n - 1

    # Проверка, что x^(2^n - 1) ≡ 1 (mod poly)

    if gf2\_poly\_powmod([1, 0], order, poly) != [1]:

        return False

    # Проверка, что условие не выполняется для любого делителя 2^n - 1

    for q in prime\_factors(order):

        if gf2\_poly\_powmod([1, 0], order // q, poly) == [1]:

            return False

    return True

for i, f in enumerate(is\_irreducible\_examples, start=1):

    print(f"Пример {i}.")

    print(f"Многочлен F(x): {f}")

    print("Проверка на примитивность:", is\_primitive(f))

    print("\n")