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 №3**

**по курсу**

**«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»**

Работу выполнил

Студенты 46 группы

Нагалевский А.М.

Преподаватель:

Крамаренко А.А.

Краснодар 2024

**Постановка задачи.**

Реализовать интерактивный калькулятор в поле Галуа с заданным пользователем образующим многочленом (сложение, умножение, деление, НОД, возведение в степень и таблица умножения).

**Поле Галуа** — это произвольное поле, состоящие из конечного числа элементов. — стандартное обозначение полей Галуа, где в скобках указывается количество элементов поля. С точностью до изоморфизма конечное поле полностью определяется его порядком, который всегда является степенью какого-нибудь простого числа, то есть , где p - простое число, а n – любое натурное число. При этом p будет являться характеристикой этого поля.

Рассмотрим также некоторые понятия, которые важны для понимания задачи.

**Неприводимый полином** — это полином, который не может быть разложен на множители меньшей степени с коэффициентами из данного поля. Например, в поле GF(2) неприводимый полином степени, n - это полином степени n без множителей меньшей степени, который не может быть представлен как произведение двух меньших по степени полиномов (рисунок 1).

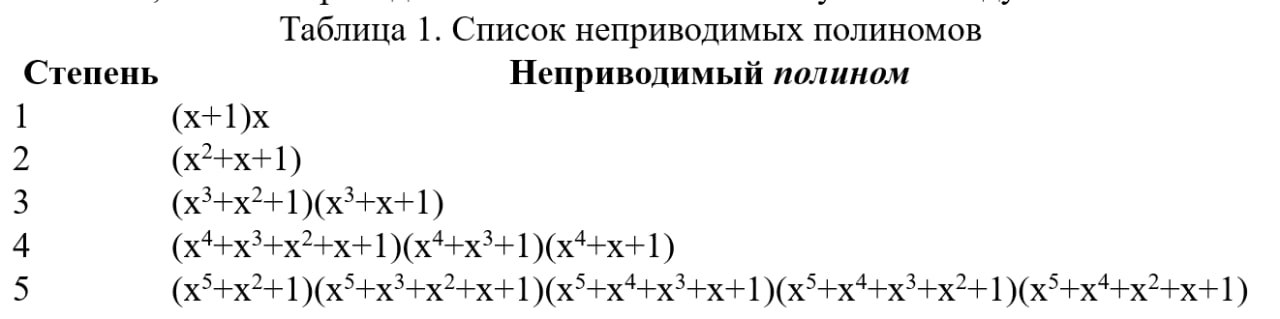


Рисунок 1 – таблица неприводимых полиномов.

**Примитивный полином** в поле Галуа — это неприводимый полином, порождающий мультипликативную группу поля. Русским языком, если все его коэффициенты взаимно просты в совокупности (НОД его коэффициентов равен 1, или по-другому: у этих чисел нет общих делителей).

**Образующий полином** в контексте конечного поля, образующий полином порождает всё поле как пространство над его базовым полем.

Как и было сказано раннее, ключевой аспект полей Галуа заключается в том, что они имеют конечное количество элементов. Это означает, что арифметические операции в этих полях выполняются по модулю числа q.

Существует два основных способа построения полей Галуа:

1. Поле Галуа GF(p), где p - простое число.
2. Расширенное поле Галуа GF(2^m), где m - положительное целое число.

Калькулятор будет совершать операции именно в расширенных полях Галуа.

Теперь рассмотрим, как выполняются операции в полях Галуа:

1. **Сложение и вычитание**: Операции сложения и вычитания в полях Галуа выполняются так же, как и в обычных полях. Результат сложения или вычитания двух элементов также должен принадлежать к этому полю.
2. **Умножение**: Умножение в полях Галуа может быть выполнено с использованием таблицы умножения, которая определена для каждого поля. Результатом будет являться умножение элементов, которые входят в это поле, друг на друга.
3. **Деление**: Деление в полях Галуа может быть выполнено как умножение на обратный элемент. Обратный элемент для каждого элемента также определен в поле Галуа.
4. **Возведение в степень**: Возведение в степень в полях Галуа может быть выполнено путем многократного умножения, которое мы уже разобрали выше.
5. **Наибольший общий делитель (НОД)**: Для полей Галуа с простым числом элементов вычисление НОД может быть выполнено аналогично алгоритму Евклида для целых чисел только с учетом степени поля.

Далее будут представлены примеры работы калькулятора (рисунки 2-4).

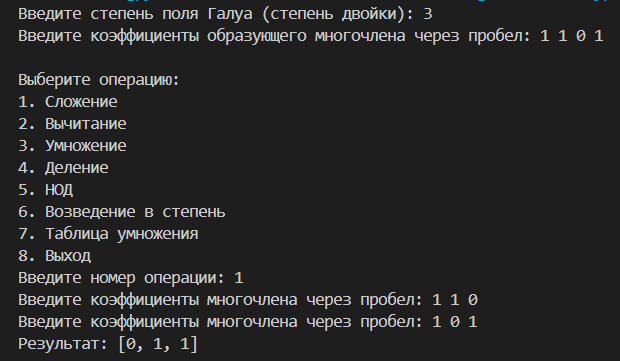


Рисунок 2 – Сложение.

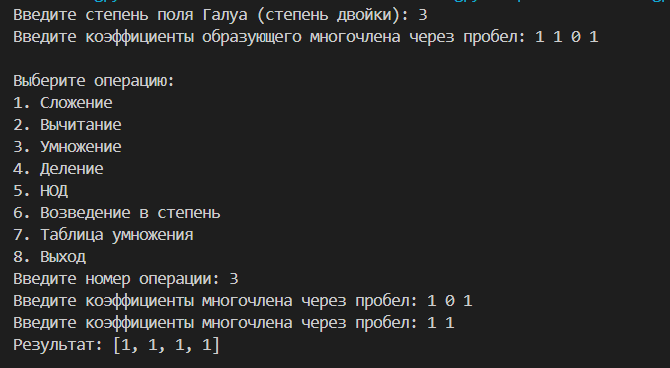


Рисунок 3 – Умножение.

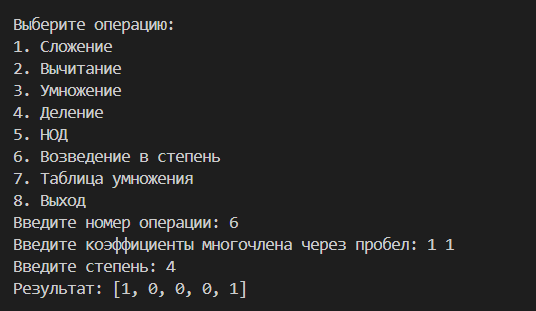


Рисунок 4 – Возведение в степень.

**Текст программы:**

**Файл LR3.py:**

class GaloisFieldCalculator:

    def \_\_init\_\_(self, modulus\_poly, coeffs):

        self.modulus\_poly = modulus\_poly

        self.coeffs = coeffs

    def \_\_add\_\_(self, other):

        result\_coeffs = [(self.coeffs[i] + other.coeffs[i]) % 2 for i in range(max(len(self.coeffs), len(other.coeffs)))]

        return GaloisFieldCalculator(self.modulus\_poly, result\_coeffs)

    def \_\_mul\_\_(self, other):

        result\_coeffs = [0] \* (len(self.coeffs) + len(other.coeffs) - 1)

        for i in range(len(self.coeffs)):

            for j in range(len(other.coeffs)):

                result\_coeffs[i+j] += self.coeffs[i] \* other.coeffs[j]

        for i in range(len(result\_coeffs)):

            result\_coeffs[i] %= 2

        return GaloisFieldCalculator(self.modulus\_poly, result\_coeffs)

    def \_\_sub\_\_(self, other):

        result\_coeffs = [(self.coeffs[i] - other.coeffs[i]) % 2 for i in range(max(len(self.coeffs), len(other.coeffs)))]

        return GaloisFieldCalculator(self.modulus\_poly, result\_coeffs)

    def \_\_truediv\_\_(self, other):

        q = GaloisFieldCalculator(self.modulus\_poly, [0])

        r = self

        while len(r.coeffs) >= len(other.coeffs):

            leading\_degree = len(r.coeffs) - 1

            other\_degree = len(other.coeffs) - 1

            q += GaloisFieldCalculator(self.modulus\_poly, [0]\*(leading\_degree - other\_degree) + [1])

            r -= (GaloisFieldCalculator(self.modulus\_poly, [0]\*(leading\_degree - other\_degree) + [1]) \* other)

        return q, r

    def gcd(self, other):

        a, b = self, other

        while b != GaloisFieldCalculator(self.modulus\_poly, [0]):

            a, b = b, a % b

        return a

    def \_\_pow\_\_(self, power):

        result = GaloisFieldCalculator(self.modulus\_poly, [1])

        for \_ in range(power):

            result \*= self

        return result

    def multiplication\_table(self):

        for i in range(1, len(self.coeffs)):

            for j in range(1, len(self.coeffs)):

                result = GaloisFieldCalculator(self.modulus\_poly, [0]\*(i+j-1)+[1]) \* GaloisFieldCalculator(self.modulus\_poly, [0]\*(i+j-1)+[1])

                print(f"{i} \* {j} = {result.coeffs}")

def input\_GaloisFieldCalculator(modulus\_poly):

    coeffs = list(map(int, input("Введите коэффициенты многочлена через пробел: ").split()))

    return GaloisFieldCalculator(modulus\_poly, coeffs)

def is\_irreducible(poly):

    for i in range(1, len(poly) // 2 + 1):

        if len(poly) % i == 0:

            factors = [poly[j:j+i] for j in range(0, len(poly), i)]

            if all(factor == factors[0] for factor in factors):

                return False

    return True

def input\_modulus\_poly():

    while True:

        coeffs = list(map(int, input("Введите коэффициенты образующего многочлена через пробел: ").split()))

        if is\_irreducible(coeffs):

            return coeffs

        else:

            print("Многочлен не является неприводимым. Пожалуйста, введите неприводимый многочлен.")

def main():

    field\_degree = int(input("Введите степень поля Галуа (степень двойки): "))

    modulus\_poly\_coeffs = input\_modulus\_poly() \* field\_degree

    modulus\_poly = GaloisFieldCalculator(None, modulus\_poly\_coeffs)

    while True:

        print("\nВыберите операцию:")

        print("1. Сложение")

        print("2. Вычитание")

        print("3. Умножение")

        print("4. Деление")

        print("5. НОД")

        print("6. Возведение в степень")

        print("7. Таблица умножения")

        print("8. Выход")

        choice = int(input("Введите номер операции: "))

        if choice == 1:

            poly1 = input\_GaloisFieldCalculator(modulus\_poly)

            poly2 = input\_GaloisFieldCalculator(modulus\_poly)

            result = poly1 + poly2

            print("Результат:", result.coeffs)

        elif choice == 2:

            poly1 = input\_GaloisFieldCalculator(modulus\_poly)

            poly2 = input\_GaloisFieldCalculator(modulus\_poly)

            result = poly1 - poly2

            print("Результат:", result.coeffs)

        elif choice == 3:

            poly1 = input\_GaloisFieldCalculator(modulus\_poly)

            poly2 = input\_GaloisFieldCalculator(modulus\_poly)

            result = poly1 \* poly2

            print("Результат:", result.coeffs)

        elif choice == 4:

            poly1 = input\_GaloisFieldCalculator(modulus\_poly)

            poly2 = input\_GaloisFieldCalculator(modulus\_poly)

            quotient, remainder = poly1 / poly2

            print("Частное:", quotient.coeffs)

            print("Остаток:", remainder.coeffs)

        elif choice == 5:

            poly1 = input\_GaloisFieldCalculator(modulus\_poly)

            poly2 = input\_GaloisFieldCalculator(modulus\_poly)

            gcd = poly1.gcd(poly2)

            print("НОД:", gcd.coeffs)

        elif choice == 6:

            poly = input\_GaloisFieldCalculator(modulus\_poly)

            power = int(input("Введите степень: "))

            result = poly \*\* power

            print("Результат:", result.coeffs)

        elif choice == 7:

            poly = input\_GaloisFieldCalculator(modulus\_poly)

            poly.multiplication\_table()

        elif choice == 8:

            print("Выход...")

            break

        else:

            print("Неверный выбор операции.")

            continue

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

    main()