**СПЕЦІАЛЬНІ РОЗДІЛИ**

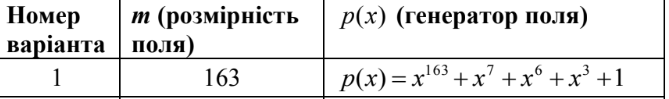
**ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ**

КОМП’ЮТЕРНИЙ ПРАКТИКУМ No3

**Реалізація операцій у скінченних полях характеристики 2**

**(поліноміальний базис)**

*ФБ-11 Данькова Єлізавета  
Варіант 1*



Реалізувати поле Галуа характеристики 2 степеня m в поліноміальному базисі з операціями:

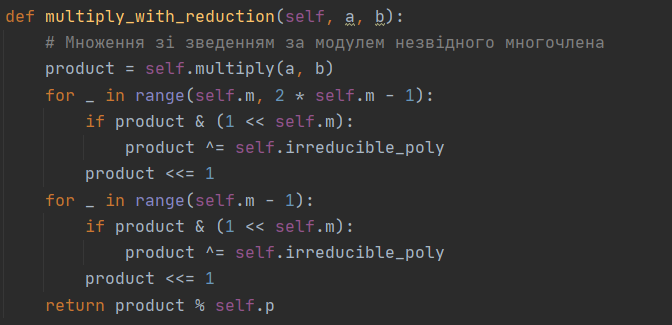
1) знаходження константи 0 – нейтрального елемента по операції «+»;

2) знаходження константи 1 – нейтрального елемента по операції «»;

3) додавання елементів;

def add(self, a, b):  
 # Додавання за модулем p  
 result = a ^ b  
 return result % self.p

4) множення елементів;



5) обчислення сліду елементу;

def trace(self, a):  
 # Слід елемента поля  
 result = a  
 for \_ in range(1, self.m):  
 a = self.multiply(a, a)  
 result ^= a  
 return result % self.p

6) піднесення елемента поля до квадрату;

def square(self, a):  
 # Піднесення до квадрату за модулем незвідного многочлена  
 a\_sq = 0  
 for i in range(self.m):  
 if a & (1 << i):  
 a\_sq ^= (1 << (2 \* i))  
 a\_sq ^= a  
 for \_ in range(self.m - 1):  
 if a\_sq & (1 << self.m):  
 a\_sq ^= self.irreducible\_poly  
 a\_sq <<= 1  
 return a\_sq % self.p

7) піднесення елемента поля до довільного степеня (не вище 2^m-1, де m –розмірність розширення);

def power(self, a, n):  
 # Піднесення до степеня  
 result = 1  
 while n > 0:  
 if n & 1:  
 result = self.multiply(result, a)  
 a = self.square(a)  
 n >>= 1  
 return result % self.p

8) знаходження оберненого елемента за множенням;

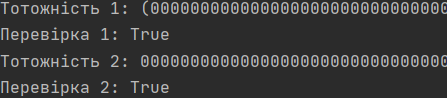
def inverse\_with\_extended\_euclidean(self, a):  
 # Обчислення оберненого елемента за допомогою розширеного алгоритму Евкліда  
 t, new\_t = 0, 1  
 r, new\_r = self.p, a  
  
 while new\_r != 0:  
 quotient = r // new\_r  
 t, new\_t = new\_t, t - quotient \* new\_t  
 r, new\_r = new\_r, r - quotient \* new\_r  
  
 if r > 1:  
 raise ValueError(f"{a} is not invertible")  
 if t < 0:  
 t += self.p  
  
 return t % self.p

9) конвертування (переведення) елемента поля в m -бітний рядок (строкове зображення) і навпаки, де m – розмірність розширення;

def to\_binary\_string(self, a):  
 # Конвертація елемента поля в m-бітний рядок  
 return bin(a)[2:].zfill(self.m)  
  
def from\_binary\_string(self, s):  
 # Конвертація m-бітного рядка в елемент поля  
 return int(s, 2)

Б) Проконтролювати коректність реалізації поля для кожної операції;

# Тотожність (a + b) \* c = b \* c + c \* a  
left\_side = field.multiply(field.add(a, b), c)  
right\_side = field.add(field.multiply(b, c), field.multiply(c, a))  
print(f"Тотожність 1: ({field.to\_binary\_string(a)} + {field.to\_binary\_string(b)}) \* {field.to\_binary\_string(c)} = {field.to\_binary\_string(left\_side)}")  
print(f"Перевірка 1: {left\_side == right\_side}")  
  
# Тотожність d^(2^m-1) = 1 для деякого d  
left\_side = field.power(d, (1 << m) - 1)  
right\_side = field.power(d, (1 << m) - 1)  
print(f"Тотожність 2: {field.to\_binary\_string(d)}^(2^m-1) = {field.to\_binary\_string(left\_side)}")  
print(f"Перевірка 2: {left\_side == right\_side}")



В) 