Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Фізико-технічний інститут

**ЛАБОРАТОРНА РОБОТА №4**

**з дисципліни**

**«Криптографія»**

**на тему:**

# «Побудова регістрів зсуву з лінійним зворотним зв’язком та дослідження їх властивостей»

Виконала:

студентка 3 курсу ФТІ

групи ФБ-73

Божко Анастасія

Перевірили:

Чорний О.

Савчук М. М.

Завадська Л. О.

# Варіант 5

**Мета роботи**

Ознайомлення з принципами побудови регістрів зсуву з лінійним зворотним зв'язком; практичне освоєння їх програмної реалізації; дослідження властивостей лінійних рекурентних послідовностей та їх залежності від властивостей характеристичного полінома регістра.

**Порядок виконання роботи**

0. Уважно прочитати методичні вказівки до виконання комп'ютерного практикуму.
1. Вибрати свій варіант завдання згідно зі списком. Варіанти завдань містяться у файлі Crypto_CP4 LFSR_Var.
2. За даними характеристичними многочленами p1(x), p2(x) скласти лінійні рекурентні співвідношення для ЛРЗ, що задаються цими характеристичними многочленами.
3. Написати програми роботи кожного з ЛРЗ L1 , L2 .
4. За допомогою цих програм згенерувати імпульсні функції для кожного з ЛРЗ і підрахувати їх періоди.
5. За отриманими результатами зробити висновки щодо влавстивостей кожного з характеристичних многочленів p1(x), p2(x): многочлен примітивний над F2 ; не примітивний, але може бути незвідним; звідний.
6. Для кожної з двох імпульсних функцій обчислити розподіл k-грам на періоді, k≤ni, де ni - степінь полінома fi(x), i=1,2 а також значення функції автокореляції A(d) для 0≤ d ≤ 10. За результатами зробити висновки.

**$P_1(X)= X_{20}+X_{17}+X_{15}+X_{14}+X_9+X_7+X_5+X_3+X_2+X+1$**

Period 1048575 примітивний

Автокореляція:

d=1: 524288
d=2: 524288
d=3: 524288
d=4: 524288
d=5: 524288
d=6: 524288
d=7: 524288
d=8: 524288
d=9: 524288
d=10: 524288

| 1gram | 2gram | 3gram | 4gram |
|---|---|---|---|
| 0 524287 | 00 262143 | 000 131071 | 0000 65535 |
| 1 524288 | 01 262144 | 001 131072 | 0001 65536 |
| | 10 262143 | 010 131072 | 0010 65536 |
| | 11 262144 | 011 131072 | 0011 65536 |
| | | 100 131071 | 0100 65536 |
| | | 101 131072 | 0101 65536 |
| | | 110 131071 | 0110 65535 |
| | | 111 131072 | 0111 65536 |
| | | | 1000 65535 |
| | | | 1001 65536 |
| | | | 1010 65536 |
| | | | 1011 65536 |
| | | | 1100 65535 |
| | | | 1101 65536 |
| | | | 1110 65536 |
| | | | 1111 65536 |

**$P_2(X)= X_{24} + X_{22} + X_{18} + X_{17} + X_{16} + X_{15} + X_{12} + X_{11} + X_9 + X_4 + X_2 + X+1$**

Period 1118481 звідний

Автокореляція:

d=1: 559680
d=2: 559392
d=3: 559392
d=4: 558432
d=5: 560000
d=6: 559488
d=7: 559488
d=8: 559392
d=9: 558432
d=10: 559488

| 1gram | 2gram | 3gram | 4gram |
|---|---|---|---|
| 0 560049 | 00 280209 | 000 140361 | 0000 70197 |
| 1 558432 | 01 279840 | 001 139848 | 0001 70164 |
| | 10 279839 | 010 139992 | 0010 69972 |
| | 11 278592 | 011 139848 | 0011 69876 |
| | | 100 139847 | 0100 69972 |
| | | 101 139992 | 0101 70020 |
| | | 110 139847 | 0110 70211 |
| | | 111 138744 | 0111 69636 |
| | | | 1000 70163 |
| | | | 1001 69684 |
| | | | 1010 70020 |
| | | | 1011 69972 |
| | | | 1100 69875 |
| | | | 1101 69972 |
| | | | 1110 69636 |
| | | | 1111 69108 |

**Код програми:**

```
#include "pch.h"
#include <cstdlib>
#include <vector>
#include <iostream>
#include <fstream>
#include <math.h>
using namespace std;
long long q = 1;
int period;
int equation_cur0[] = { 1,1,1,1,0,1,0,1,0,1,0,0,0,0,1,1,0,1,0,0 };
int equation_cur1[] = { 1,1,1,1,0,1,0,1,0,1,0,0,0,0,1,1,0,1,0,0 };//коэф. уравнения20+X17+X15+X14+X9+X7+X5+X3+X2+X+1
int equation_cur[]={1,1,1,0,1,0,0,0,0,1,0,1,1,0,0,1,1,1,1,1,0,0,0,1,0};//коэф.уравненияX24+X22+X18+X17+X16+X15+X12+X11+X9+X4+X2+X+1
int gram_1[] = { 0,1 };
int gram_2[] = { 0,0,0,1,1,0,1,1 };
int gram_3[] = { 0,0,0,0,0,1,0,1,0,0,1,1,1,0,0,1,0,1,1,1,0,1,1,1 };
int gram_4[] = { 0,0,0,0,0,0,0,1,0,0,1,0,0,0,1,1,0,1,0,0,0,1,0,1,0,1,1,0,0,1,1,1,1,1,0,0,0,1,0,0,1,1,0,1,0,1,0,1,0,1,1,1,1,1,0,0,1,1,0,1,1,1,1,1,0,1,1,1,1 };
int size_eq = sizeof(equation_cur) / sizeof(equation_cur[0]);
vector<int> shift((pow(2, size_eq) - 1));//сдвиг регистра(первый элемент)
vector<int> recur_sequence((pow(2, size_eq) - 1) * size_eq);//рекурсивная последовательность
void find_recur_sequence() {// выводит
	cout << size_eq << endl;
	for (long long i = 0; i < (pow(2, size_eq) - 1); i++) {
			for (int n = 0; n < size_eq; n++) {
			if (i > 0 && n != (size_eq - 1)) {//сдвигаем на элемент вперед по модулю size_eq
				recur_sequence[i * size_eq + n] = recur_sequence[(i - 1) * size_eq + n + 1];}
			if (((i + n) >= size_eq) && (n == size_eq - 1)) {
				recur_sequence[i * size_eq + n] = 0;
				for (int j = n; j >= 0; j--) {
	recur_sequence[i*size_eq+n]=(recur_sequence[i*size_eq+ n] + recur_sequence[(i - 1)* size_eq + j]*equation_cur[((i-1)*size_eq+j)%size_eq])%2;
			while (recur_sequence[i * size_eq + n] < 0)recur_sequence[i * size_eq + n] = recur_sequence[i * size_eq + n] + 2;
				}	}	}	} }
void find_shift() {long long n=0;
		for (long long i = 0; i < (pow(2, size_eq) - 1) * size_eq; i++) {
		shift[n] = recur_sequence[i];
		n++;
		i = i + size_eq - 1;	}}
long long period_rec() {for (long long i = 1; i < (pow(2, size_eq) - 1) ;i++) {
		int y = 0;
		for (int n = 0; n < size_eq;n++) {
			if ((recur_sequence[n] == recur_sequence[i*size_eq + n])) {////((i*q+n)< (pow(2, size_eq) - 1) * size_eq)&&
				y++;	} }
		if (y == size_eq) {	break;		}
		else {	q++; if (q > (pow(2, size_eq) - 1))break;	}	}
	return q; }
void autocorrelation() {
	long int s = (pow(2, size_eq) - 1);
	vector<long long> d(10);
	for (int n = 1; n <= 10; n++) {	d[n-1] = 0;
		for (long long i = 0; i < period; i++) {//1 1 1 1 1
			d[n-1] = d[n-1] + (shift[i]+shift[((i+n)% period)])%2;		}	}
```

```
        for (int n= 0; n < 10; n++) {//1 1 1 1 1
                cout<<" "<<d[n];   } }

int partition_1gramm(int n, int Ngramm[]) {//n=1
        for (int i = 0; i < pow(2, n)*n; i = i + n) {
                int count = 0;
                for (int j = 0; j < period; j++) {
                        if (Ngramm[i] == shift[j])count++;
                }
                cout << endl << Ngramm[i] << " " << count;    }
        return 0;}
int partition_2gramm(int n, int Ngramm[]) {//n=1
        for (int i = 0; i < pow(2, n)*n; i = i + n) {
                int count = 0;
                for (int j = 1; j < period; j++) {
                        if ((Ngramm[i] == shift[j - 1]) && (Ngramm[i + 1] == shift[j]))count++;            }
                cout << endl << Ngramm[i] << Ngramm[i + 1] << " " << count;      }
        return 0;}
int partition_3gramm(int n, int Ngramm[]) {//n=1
        for (int i = 0; i < pow(2, n)*n; i = i + n) {
                int count = 0;
                for (int j = 2; j < period; j++) {
                        if ((Ngramm[i] == shift[j - 2]) && (Ngramm[i + 1] == shift[j - 1]) && (Ngramm[i + 2] == shift[j]))count++;              }
                cout << endl << Ngramm[i] << Ngramm[i + 1] << Ngramm[i + 2] << " " << count;       }
        return 0;}
int partition_4gramm(int n, int Ngramm[]) {//n=1
        for (int i = 0; i < pow(2, n)*n; i = i + n) {
                int count = 0;
                for (int j = 3; j < period; j++) {
                        if ((Ngramm[i] == shift[j - 3]) && (Ngramm[i + 1] == shift[j - 2]) && (Ngramm[i + 2] == shift[j - 1]) && (Ngramm[i + 3]
== shift[j]))count++;
                }cout << endl << Ngramm[i] << Ngramm[i + 1] << Ngramm[i + 2] << Ngramm[i + 3] << " " << count;       }
        return 0;}

int main(){
        cout << size_eq << " ";
        for (int i = 0; i < size_eq; i++) {//находим первые size_eq элементы
                if(i==(size_eq-1)) recur_sequence[i] = 1;
                else recur_sequence[i] = 0; }
        find_recur_sequence();
         cout << endl;
        period = period_rec();
        cout << "Period " << period<<endl;
        find_shift();
        ofstream f;
        //открываем файл в режиме записи,
        //режим ios::out устанавливается по умолчанию
        f.open("C:\\Users\\Настя\\Desktop\\l.txt", ios::out);
        //цикл для ввода вещественных чисел
        //и записи их в файл
        for (long long i = 0; i < (pow(2, size_eq) - 1); i++) {
                //cout << shift[n];
                f << shift[i] << " ";
                //cout << shift[i];
                //закрытие потока
        }
        f.close();
        autocorrelation();
        partition_1gramm(1, gram_1);
        partition_2gramm(2, gram_2);
        partition_3gramm(3, gram_3);
        partition_4gramm(4, gram_4);
        return 0;}
```

## Висновок:

В даному комп'ютерному практикумі було набуто навичок роботи з лінійними регістрами зсуву, а саме: їх програмна реалізація, дослідження властивостей характеристичного полінома регістра. Окрім цього було досліджено властивості лінійних рекурентних послідовностей.