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»

Виконали:
студентки 3 курсу ФТІ
групи ФБ-71
Гресь В. Нацвін К.
Перевірили:
Чорний О.
Савчук М. М.
Завадська Л. О.

Київ 2019

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.

Рекурентні співвідношення:

6	$P_1(X)= X^{22}+X^{17}+ X^{15}+X^{14}+X^{13}+X^{11}+X^8+X^7+X^6+X^3+X^2+X+1$
	$P_2(X)= X^{21}+X^{16}+X^{15}+X^{14}+X^{13}+X^{10}+X^9+X^7+X^2+X+1$

Результати:
Поліном p1 = [1,1,1,1,0,0,1,1,1,0,0,1,0,1,1,1,0,1,0,0,0,0]
Період: 584073
Значення автокереляції
autocorrelation1: 292040 autocorrelation2: 292040 autocorrelation3: 292028 autocorrelation4: 292040
autocorrelation5: 292040 autocorrelation6: 292036 autocorrelation7: 292028 autocorrelation8: 292040
autocorrelation9: 292040 autocorrelation10: 292028
N – грами:

Биграммы:	1000 8998	01010 3615
00 73131	1001 8975	10111 3680
01 72863	1010 9200	00011 3658
10 72908	0110 9143	10000 3685
11 73134	1011 9120	10110 3628
Триграммы:	1110 9139	11010 3656
000 24317	0001 9155	10101 3677
100 24354	1101 9100	01111 3594
001 24340	0101 9123	11110 3756
011 24323	0111 9037	11011 3679
111 24340	1100 9272	01110 3636
010 24354	0011 9186	11111 3602
101 24324	**Пятиграмы**:	00111 3581
110 24338	00000 3615	11101 3657
Четыреграммы:	01000 3660	01001 3609
0000 9090	01011 3652	00100 3545
0100 9207	11100 3675	01101 3597
0010 9133	00010 3674	10010 3707
1111 9140	10011 3734	11001 3563

00110 3613
10001 3552
10100 3732
01100 3679
11000 3658
00101 3712
00001 3733

Поліном p2 = [1,1,1,0,0,0,0,1,0,1,1,0,0,1,1,1,1,0,0,0,0]
Період: 2097151
Значення автокереляції
autocorrelation1: 1048576 autocorrelation2: 1048576 autocorrelation3: 1048576 autocorrelation4: 1048576
autocorrelation5: 1048576 autocorrelation6: 1048576 autocorrelation7: 1048576 autocorrelation8: 1048576
autocorrelation9: 1048576 autocorrelation10: 1048576
N – грами:
Биграммы:
00 262208
10 262034
01 262125
11 262208
Триграммы:
000 87279
001 87491
011 87292
110 86972
100 87579
101 87626
111 87457
010 87354
Четыреграммы:
0000 32850
1000 32659
0111 32853
1011 32982
0001 32798
1001 32681
1100 32595
0110 32546
0101 32851
1111 32661
1110 32646
1101 32799
1010 32910
0010 32700
0100 32745
0011 33011
Пятиграммы:
00000 13139
10000 13108
11110 13201
11000 13065
10001 13045
00001 13080
01100 13043
01011 12992
11001 13136
11010 13062
01010 13316
11111 12996
00110 13086
11100 13090
00011 13205
10010 12986
01000 13171
01110 13112
10101 13005
00111 13236
00100 13073
01001 13184
00010 12976
11011 13072
11101 13028
10100 12953
10011 13163
00101 13205
10110 13248
01111 13190
01101 13141
10111 13123

Код програми:

```
#include <iostream>
#include <windows.h>
#include <string>
#include <vector>

std::string biGram[4] = {"00", "01", "10", "11"};
int biCount[4] = {0};
std::string threeGram [8] = {"000", "001", "010", "011",
"100", "101", "110", "111"};
int threeCount[8] = {0};
std::string fourGram [16] = {"0000", "0001", "0010",
"0011", "0100", "0101", "0110", "0111", "1000", "1001",
"1010", "1011", "1100", "1101", "1110", "1111"};
int fourCount[16] = {0};
std::string fiveGram [32] = {"00000", "00001", "00010",
"00011", "00100", "00101", "00110", "00111", "01000",
"01001", "01010", "01011", "01100", "01101", "01110",
"01111", "10000", "10001", "10010",  "10011", "10100",
"10101", "10110", "10111", "11000", "11001", "11010",
"11011", "11100", "11101", "11110", "11111"};
int fiveCount[32] = {0};

std::vector<bool> sequence1
{1,1,1,1,0,0,1,1,1,0,0,1,0,1,1,1,0,1,0,0,0,0};
std::vector<bool> impulse1
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1};
std::vector<bool> etalon1
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1};
std::vector<bool> sequence2
{1,1,1,0,0,0,0,1,0,1,1,0,0,1,1,1,1,0,0,0,0};
std::vector<bool> impulse2
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1};
std::vector<bool> etalon2
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1};

int doEverything(std::vector<bool> sequence,
std::vector<bool> impulse, std::vector<bool> etalon)
{
    std::vector<bool> order, temp;
    bool sum;

    do
    {
        order.push_back(sequence[1]);
```

```
    for (int i = 0; i < sequence.size(); ++i)
    {
        temp.push_back(impulse[i] & sequence[i]);
    }

    sum = 0;
    for (int i = 0; i < sequence.size(); ++i)
    {
        sum ^= temp[i];
    }

    impulse.erase (impulse.begin());
    impulse.push_back(sum);

    for (int i = 0; i < sequence.size(); ++i)
    {
        temp.pop_back();
    }

} while (!(impulse == etalon));

std::cout << order.size();

for (int i = 0; i < 4; ++i)
{
    for (int j = 0; j < order.size(); j += 2)
    {
        std::string buf(sequence[j], sequence[j + 1]);
        if (buf == biGram[i])
        {
            biCount[i]++;
        }
    }
    std::cout << biCount[i];
}

for (int i = 0; i < 8; ++i)
{
    for (int j = 0; j < order.size(); j += 3)
    {
        std::string buf(sequence[j], sequence[j + 2]);
        if (buf == threeGram[i])
        {
            threeCount[i]++;
        }
    }
    std::cout << threeCount[i];
}

for (int i = 0; i < 16; ++i)
{
    for (int j = 0; j < order.size(); j += 4)
    {
        std::string buf(sequence[j], sequence[j + 3]);
        if (buf == fourGram[i])
        {
            fourCount[i]++;
        }
    }
    std::cout << fourCount[i];
}

    for (int i = 0; i < 32; ++i)
    {
        for (int j = 0; j < order.size(); j += 5)
        {
            std::string buf(sequence[j], sequence[j + 4]);
            if (buf == fiveGram[i])
            {
                fiveCount[i]++;
            }
        }
        std::cout << fiveCount[i];
    }

    for (int j = 0; j < 10; j++)
    {
        int autocorrelation = 0;
        for (int i = 0; i < order.size(); i += j)
        {
            autocorrelation += order[i] + order[i + j] % 2;
        }
        std::cout << "\nautocorrelation " << j << ' ' <<
autocorrelation;
    }
}

int main()
{
    doEverything(sequence1, impulse1, etalon1);
    doEverything(sequence2, impulse2, etalon2);
}
```