МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Криптографія

Комп'ютерний практикум №4

“Побудова регістрів зсуву з лінійним зворотнім зв'язком та дослідження їх властивостей”

Перевірив:	Виконали:
Чорний О.М.	Студенти групи ФБ-71
Завадська Л.О.	Новик Л.А.
Савчук М.М.	Равкін Д.Б.

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 $p_1(x)$, $p_2(x)$ скласти лінійні рекурентні співвідношення для ЛРЗ, що задаються цими характеристичними многочленами.
3. Написати програми роботи кожного з ЛРЗ ${}_1L$, ${}_2L$.
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 $p_1(x)$, $p_2(x)$: многочлен примітивний над ${}_2F$; не примітивний, але може бути незвідним; звідний.
6. Для кожної з двох імпульсних функцій обчислити розподіл k-грам на періоді, $k \le n_i$, де n_i - степінь полінома $f_i(x)$, i=1,2 а також значення функції автокореляції $A(d)$ для $0 \le d \le 10$. За результатами зробити висновки.

Результати:

$P_1(X) = X_{21} + X_{20} + X_{19} + X_{18} + X_{12} + X_{11} + X_{10} + X_8 + X_6 + X_5 + 1$

T для першого полінома:1755464

Автокореляція:

1:877768
2:877944
3:877860
4:877180
5:877828
6:878024
7:877346
8:877976
9:877280
10:877940

Біграми	3-грами	4-грами	5-грами	6-грами
00 219793	000 73225	0000 27583	00000 10966	000000 4610
01 219716	001 73236	0001 27359	00001 10975	000001 4602
10 218768	010 73168	0010 27262	00010 11084	000010 4637
11 219455	011 73528	0011 27661	00011 10901	000011 4630
	100 73177	0100 27444	00100 11028	000100 4495
	101 73136	0101 27548	00101 10929	000101 4603
	110 72568	0110 27561	00110 10840	000110 4457
	111 73116	0111 27364	00111 11013	000111 4637
		1000 27379	01000 10990	001000 4517
		1001 27286	01001 10939	001001 4473
		1010 27343	01010 10972	001010 4560
		1011 27214	01011 10898	001011 4572
		1100 27522	01100 10980	001100 4603
		1101 27606	01101 10925	001101 4588
		1110 27380	01110 11043	001110 4604
		1111 27354	01111 10937	001111 4522
			10000 10908	010000 4587
			10001 11086	010001 4716

			10010 10897 10011 10909 10100 11116 10101 11025 10110 11231 10111 11040 11000 11032 11001 10922 11010 10856 11011 10827 11100 10930 11101 10903 11110 11011 11111 10979	010010 4537 010011 4573 010100 4622 010101 4577 010110 4475 010111 4590 011000 4586 011001 4687 011010 4522 011011 4610 011100 4649 011101 4569 011110 4653 011111 4609 100000 4505 100001 4552 100010 4555 100011 4551 100100 4600 100101 4607 100110 4489 100111 4504 101000 4535 101001 4639 101010 4563 101011 4536 101100 4625 101101 4557 101110 4455 101111 4653 110000 4491 110001 4633 110010 4583 110011 4553 110100 4572 110101 4534 110110 4536 110111 4486 111000 4723 111001 4495 111010 4534 111011 4618 111100 4648 111101 4538 111110 4511 111111 4524

$P_2(X)= X_{23}+X_{22}+X_{21}+X_{20}+X_{19}+X_{16}+X_{15}+X_{13}+X_{12}+X_9+X_6+X_3+1$
T для другого полінома:797734
Автокореляція:
1:398862
2:398854
3:398844
4:398782

5:398832
6:398840
7:398868
8:398872
9:398878
10:398820

Біграми	3-грами	4-грами	5-грами	6-грами
00 99304	000 33000	0000 12409	00000 5015	000000 2048
01 99555	001 33254	0001 12364	00001 4992	000001 2079
10 99868	010 33029	0010 12290	00010 4749	000010 2052
11 100140	011 33171	0011 12492	00011 4913	000011 2066
	100 33296	0100 12416	00100 4969	000100 2091
	101 33421	0101 12417	00101 4955	000101 2070
	110 33281	0110 12478	00110 4933	000110 2044
	111 33459	0111 12545	00111 5028	000111 2059
		1000 12429	01000 4976	001000 2057
		1001 12510	01001 4838	001001 2095
		1010 12510	01010 4914	001010 2056
		1011 12564	01011 5042	001011 2040
		1100 12495	01100 4983	001100 2064
		1101 12408	01101 5012	001101 2107
		1110 12577	01110 5038	001110 2135
		1111 12529	01111 5100	001111 2134
			10000 4960	010000 2107
			10001 4955	010001 2061
			10010 4983	010010 2062
			10011 5061	010011 2012
			10100 5065	010100 2020
			10101 5062	010101 2086
			10110 5074	010110 2063
			10111 5036	010111 2056
			11000 4937	011000 2049
			11001 4956	011001 2052
			11010 5064	011010 2069
			11011 5003	011011 2086
			11100 5121	011100 2101
			11101 4908	011101 2036
			11110 4980	011110 2109
			11111 4924	011111 2116
				100000 2034
				100001 2046
				100010 2116
				100011 2042
				100100 2070
				100101 2084
				100110 2104
				100111 2108
				101000 2097
				101001 2091
				101010 2075
				101011 2143
				101100 2136
				101101 2081
				101110 2086
				101111 2051

				110000 2014 110001 2040 110010 2049 110011 2096 110100 2117 110101 2094 110110 2074 110111 2102 111000 2085 111001 2102 111010 2083 111011 2067 111100 2093 111101 2103 111110 2080 111111 2110

КОД

```
#include "windows.h"
#include <iostream>
#include <fstream>
#include<string>
#include<map>
#include<vector>
#include<algorithm>
#include<cmath>
#include <limits>
#include <iomanip>
#include <stdio.h>
#include <stdlib.h>
using namespace std;


void polinoms(vector <int> T1)
{
	//2-gramm количество
	map <string, int> bigram_kolvo;
	string bigrama; char symbol_1, symbol_2;
	for (int i = 0; i < T1.size() - 1; i++)
	{
		if (T1[i] == 0) { symbol_1 = '0'; }
		else { symbol_1 = '1'; }
		i++;
```

```
	if (T1[i] == 0) { symbol_2 = '0'; }
	else { symbol_2 = '1'; }
	bigrama.push_back(symbol_1);
	bigrama.push_back(symbol_2);
	auto iter = bigram_kolvo.find(bigrama);
	if (iter == bigram_kolvo.end()) { bigram_kolvo.emplace(bigrama, 1); }
	else { iter->second++; }
	bigrama.clear();
}
for (auto it = bigram_kolvo.begin(); it != bigram_kolvo.end(); ++it)
{
	cout << it->first << "  " << it->second << endl;
}
//3-gramm количество
map <string, int> trigram_kolvo;
string trigrama; char symbol_3;
for (int i = 0; i < T1.size() - 2; i++)
{
	if (T1[i] == 0) { symbol_1 = '0'; }
	else { symbol_1 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_2 = '0'; }
	else { symbol_2 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_3 = '0'; }
	else { symbol_3 = '1'; }
	trigrama.push_back(symbol_1);
	trigrama.push_back(symbol_2);
	trigrama.push_back(symbol_3);
	auto iter = trigram_kolvo.find(trigrama);
	if (iter == trigram_kolvo.end()) { trigram_kolvo.emplace(trigrama, 1); }
	else { iter->second++; }
	trigrama.clear();
}
for (auto it = trigram_kolvo.begin(); it != trigram_kolvo.end(); ++it)
{
	cout << it->first << "  " << it->second << endl;
```

```
}
//4-gramm количество
map <string, int> gram_kolvo;
string grama; char symbol_4;
for (int i = 0; i < T1.size() - 3; i++)
{
	if (T1[i] == 0) { symbol_1 = '0'; }
	else { symbol_1 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_2 = '0'; }
	else { symbol_2 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_3 = '0'; }
	else { symbol_3 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_4 = '0'; }
	else { symbol_4 = '1'; }
	grama.push_back(symbol_1);
	grama.push_back(symbol_2);
	grama.push_back(symbol_3);
	grama.push_back(symbol_4);
	auto iter = gram_kolvo.find(grama);
	if (iter == gram_kolvo.end()) { gram_kolvo.emplace(grama, 1); }
	else { iter->second++; }
	grama.clear();
}
for (auto it = gram_kolvo.begin(); it != gram_kolvo.end(); ++it)
{
	cout << it->first << "  " << it->second << endl;
}
gram_kolvo.clear();
//5-gramm количество
char symbol_5;
for (int i = 0; i < T1.size() - 4; i++)
{
	if (T1[i] == 0) { symbol_1 = '0'; }
	else { symbol_1 = '1'; }
```

```
	i++;
	if (T1[i] == 0) { symbol_2 = '0'; }
	else { symbol_2 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_3 = '0'; }
	else { symbol_3 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_4 = '0'; }
	else { symbol_4 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_5 = '0'; }
	else { symbol_5 = '1'; }
	grama.push_back(symbol_1);
	grama.push_back(symbol_2);
	grama.push_back(symbol_3);
	grama.push_back(symbol_4);
	grama.push_back(symbol_5);
	auto iter = gram_kolvo.find(grama);
	if (iter == gram_kolvo.end()) { gram_kolvo.emplace(grama, 1); }
	else { iter->second++; }
	grama.clear();
}
for (auto it = gram_kolvo.begin(); it != gram_kolvo.end(); ++it)
{
	cout << it->first << "  " << it->second << endl;
}
gram_kolvo.clear();
//6-gramm количество
char symbol_6;
for (int i = 0; i < T1.size() - 5; i++)
{
	if (T1[i] == 0) { symbol_1 = '0'; }
	else { symbol_1 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_2 = '0'; }
	else { symbol_2 = '1'; }
	i++;
```

```
	if (T1[i] == 0) { symbol_3 = '0'; }
	else { symbol_3 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_4 = '0'; }
	else { symbol_4 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_5 = '0'; }
	else { symbol_5 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_6 = '0'; }
	else { symbol_6 = '1'; }
	grama.push_back(symbol_1);
	grama.push_back(symbol_2);
	grama.push_back(symbol_3);
	grama.push_back(symbol_4);
	grama.push_back(symbol_5);
	grama.push_back(symbol_6);
	auto iter = gram_kolvo.find(grama);
	if (iter == gram_kolvo.end()) { gram_kolvo.emplace(grama, 1); }
	else { iter->second++; }
	grama.clear();
}
for (auto it = gram_kolvo.begin(); it != gram_kolvo.end(); ++it)
{
	cout << it->first << "  " << it->second << endl;
}
gram_kolvo.clear();

//7-gramm количество
char symbol_7;
for (int i = 0; i < T1.size() - 6; i++)
{
	if (T1[i] == 0) { symbol_1 = '0'; }
	else { symbol_1 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_2 = '0'; }
	else { symbol_2 = '1'; }
```

```
	i++;
	if (T1[i] == 0) { symbol_3 = '0'; }
	else { symbol_3 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_4 = '0'; }
	else { symbol_4 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_5 = '0'; }
	else { symbol_5 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_6 = '0'; }
	else { symbol_6 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_7 = '0'; }
	else { symbol_7 = '1'; }
	grama.push_back(symbol_1);
	grama.push_back(symbol_2);
	grama.push_back(symbol_3);
	grama.push_back(symbol_4);
	grama.push_back(symbol_5);
	grama.push_back(symbol_6);
	grama.push_back(symbol_7);
	auto iter = gram_kolvo.find(grama);
	if (iter == gram_kolvo.end()) { gram_kolvo.emplace(grama, 1); }
	else { iter->second++; }
	grama.clear();
}
for (auto it = gram_kolvo.begin(); it != gram_kolvo.end(); ++it)
{
	cout << it->first << "  " << it->second << endl;
}
gram_kolvo.clear();

//8-gramm количество
char symbol_8;
for (int i = 0; i < T1.size() - 7; i++)
{
```

```
        if (T1[i] == 0) { symbol_1 = '0'; }
        else { symbol_1 = '1'; }
        i++;
        if (T1[i] == 0) { symbol_2 = '0'; }
        else { symbol_2 = '1'; }
        i++;
        if (T1[i] == 0) { symbol_3 = '0'; }
        else { symbol_3 = '1'; }
        i++;
        if (T1[i] == 0) { symbol_4 = '0'; }
        else { symbol_4 = '1'; }
        i++;
        if (T1[i] == 0) { symbol_5 = '0'; }
        else { symbol_5 = '1'; }
        i++;
        if (T1[i] == 0) { symbol_6 = '0'; }
        else { symbol_6 = '1'; }
        i++;
        if (T1[i] == 0) { symbol_7 = '0'; }
        else { symbol_7 = '1'; }
        i++;
        if (T1[i] == 0) { symbol_8 = '0'; }
        else { symbol_8 = '1'; }
        grama.push_back(symbol_1);
        grama.push_back(symbol_2);
        grama.push_back(symbol_3);
        grama.push_back(symbol_4);
        grama.push_back(symbol_5);
        grama.push_back(symbol_6);
        grama.push_back(symbol_7);
        grama.push_back(symbol_8);
        auto iter = gram_kolvo.find(grama);
        if (iter == gram_kolvo.end()) { gram_kolvo.emplace(grama, 1); }
        else { iter->second++; }
        grama.clear();
    }
    for (auto it = gram_kolvo.begin(); it != gram_kolvo.end(); ++it)
```

```
{
	cout << it->first << "  " << it->second << endl;
}
gram_kolvo.clear();
//9-gramm количество
char symbol_9;
for (int i = 0; i < T1.size() - 8; i++)
{
	if (T1[i] == 0) { symbol_1 = '0'; }
	else { symbol_1 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_2 = '0'; }
	else { symbol_2 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_3 = '0'; }
	else { symbol_3 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_4 = '0'; }
	else { symbol_4 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_5 = '0'; }
	else { symbol_5 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_6 = '0'; }
	else { symbol_6 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_7 = '0'; }
	else { symbol_7 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_8 = '0'; }
	else { symbol_8 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_9 = '0'; }
	else { symbol_9 = '1'; }
	grama.push_back(symbol_1);
	grama.push_back(symbol_2);
	grama.push_back(symbol_3);
```

```
	grama.push_back(symbol_4);
	grama.push_back(symbol_5);
	grama.push_back(symbol_6);
	grama.push_back(symbol_7);
	grama.push_back(symbol_8);
	grama.push_back(symbol_9);
	auto iter = gram_kolvo.find(grama);
	if (iter == gram_kolvo.end()) { gram_kolvo.emplace(grama, 1); }
	else { iter->second++; }
	grama.clear();
}
for (auto it = gram_kolvo.begin(); it != gram_kolvo.end(); ++it)
{
	cout << it->first << "  " << it->second << endl;
}
gram_kolvo.clear();
//10-gramm количество
char symbol_10;
for (int i = 0; i < T1.size() - 9; i++)
{
	if (T1[i] == 0) { symbol_1 = '0'; }
	else { symbol_1 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_2 = '0'; }
	else { symbol_2 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_3 = '0'; }
	else { symbol_3 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_4 = '0'; }
	else { symbol_4 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_5 = '0'; }
	else { symbol_5 = '1'; }
	i++;
	if (T1[i] == 0) { symbol_6 = '0'; }
	else { symbol_6 = '1'; }
```

```
            i++;
            if (T1[i] == 0) { symbol_7 = '0'; }
            else { symbol_7 = '1'; }
            i++;
            if (T1[i] == 0) { symbol_8 = '0'; }
            else { symbol_8 = '1'; }
            i++;
            if (T1[i] == 0) { symbol_9 = '0'; }
            else { symbol_9 = '1'; }
            i++;
            if (T1[i] == 0) { symbol_10 = '0'; }
            else { symbol_10 = '1'; }
            grama.push_back(symbol_1);
            grama.push_back(symbol_2);
            grama.push_back(symbol_3);
            grama.push_back(symbol_4);
            grama.push_back(symbol_5);
            grama.push_back(symbol_6);
            grama.push_back(symbol_7);
            grama.push_back(symbol_8);
            grama.push_back(symbol_9);
            grama.push_back(symbol_10);
            auto iter = gram_kolvo.find(grama);
            if (iter == gram_kolvo.end()) { gram_kolvo.emplace(grama, 1); }
            else { iter->second++; }
            grama.clear();
      }
      for (auto it = gram_kolvo.begin(); it != gram_kolvo.end(); ++it)
      {
            cout << it->first << "  " << it->second << endl;
      }
      gram_kolvo.clear();
      ////////////////////////////////////////////////////////////////////////////////
////////////////////////////karelyaciya
      int sum_ka = 0;
      for (int i = 0; i < T1.size(); i++)
      {
```

```
	if (i == T1.size() - 1) { sum_ka += (T1[i] + T1[0]) % 2; break; }
	sum_ka += (T1[i] + T1[i + 1]) % 2;

}
cout << "1:" << sum_ka << endl;
sum_ka = 0;
for (int i = 0; i < T1.size(); i++)
{
	if (i == T1.size() - 2) {
		sum_ka += (T1[i] + T1[0]) % 2;
		sum_ka += (T1[i + 1] + T1[1]) % 2; break;
	}
	sum_ka += (T1[i] + T1[i + 2]) % 2;

}
cout << "2:" << sum_ka << endl;
sum_ka = 0;
for (int i = 0; i < T1.size(); i++)
{
	if (i == T1.size() - 3) {
		sum_ka += (T1[i] + T1[0]) % 2;
		sum_ka += (T1[i + 1] + T1[1]) % 2;
		sum_ka += (T1[i + 2] + T1[2]) % 2; break;
	}

	sum_ka += (T1[i] + T1[i + 3]) % 2;

}
cout << "3:" << sum_ka << endl;
sum_ka = 0;
for (int i = 0; i < T1.size(); i++)
{
	if (i == T1.size() - 4) {
		sum_ka += (T1[i] + T1[0]) % 2;
		sum_ka += (T1[i + 1] + T1[1]) % 2;
		sum_ka += (T1[i + 2] + T1[2]) % 2;
		sum_ka += (T1[i + 3] + T1[3]) % 2; break;
```

```
	}

	sum_ka += (T1[i] + T1[i + 4]) % 2;

}
cout << "4:" << sum_ka << endl;
sum_ka = 0;
for (int i = 0; i < T1.size(); i++)
{
	if (i == T1.size() - 5) {
		sum_ka += (T1[i] + T1[0]) % 2;
		sum_ka += (T1[i + 1] + T1[1]) % 2;
		sum_ka += (T1[i + 2] + T1[2]) % 2;
		sum_ka += (T1[i + 3] + T1[3]) % 2;
		sum_ka += (T1[i + 4] + T1[4]) % 2; break;
	}

	sum_ka += (T1[i] + T1[i + 5]) % 2;

}
cout << "5:" << sum_ka << endl;
sum_ka = 0;
for (int i = 0; i < T1.size(); i++)
{
	if (i == T1.size() - 6) {
		sum_ka += (T1[i] + T1[0]) % 2;
		sum_ka += (T1[i + 1] + T1[1]) % 2;
		sum_ka += (T1[i + 2] + T1[2]) % 2;
		sum_ka += (T1[i + 3] + T1[3]) % 2;
		sum_ka += (T1[i + 4] + T1[4]) % 2;
		sum_ka += (T1[i + 5] + T1[5]) % 2; break;
	}

	sum_ka += (T1[i] + T1[i + 6]) % 2;

}
cout << "6:" << sum_ka << endl;
```

```
sum_ka = 0;
for (int i = 0; i < T1.size(); i++)
{
	if (i == T1.size() - 7) {
		sum_ka += (T1[i] + T1[0]) % 2;
		sum_ka += (T1[i + 1] + T1[1]) % 2;
		sum_ka += (T1[i + 2] + T1[2]) % 2;
		sum_ka += (T1[i + 3] + T1[3]) % 2;
		sum_ka += (T1[i + 4] + T1[4]) % 2;
		sum_ka += (T1[i + 5] + T1[5]) % 2;
		sum_ka += (T1[i + 6] + T1[6]) % 2; break;
	}

	sum_ka += (T1[i] + T1[i + 7]) % 2;

}
cout << "7:" << sum_ka << endl;
sum_ka = 0;
for (int i = 0; i < T1.size(); i++)
{
	if (i == T1.size() - 8) {
		sum_ka += (T1[i] + T1[0]) % 2;
		sum_ka += (T1[i + 1] + T1[1]) % 2;
		sum_ka += (T1[i + 2] + T1[2]) % 2;
		sum_ka += (T1[i + 3] + T1[3]) % 2;
		sum_ka += (T1[i + 4] + T1[4]) % 2;
		sum_ka += (T1[i + 5] + T1[5]) % 2;
		sum_ka += (T1[i + 6] + T1[6]) % 2;
		sum_ka += (T1[i + 7] + T1[7]) % 2; break;
	}

	sum_ka += (T1[i] + T1[i + 8]) % 2;

}
cout << "8:" << sum_ka << endl;
```

```
sum_ka = 0;
for (int i = 0; i < T1.size(); i++)
{
	if (i == T1.size() - 9) {
		sum_ka += (T1[i] + T1[0]) % 2;
		sum_ka += (T1[i + 1] + T1[1]) % 2;
		sum_ka += (T1[i + 2] + T1[2]) % 2;
		sum_ka += (T1[i + 3] + T1[3]) % 2;
		sum_ka += (T1[i + 4] + T1[4]) % 2;
		sum_ka += (T1[i + 5] + T1[5]) % 2;
		sum_ka += (T1[i + 6] + T1[6]) % 2;
		sum_ka += (T1[i + 7] + T1[7]) % 2;
		sum_ka += (T1[i + 8] + T1[8]) % 2; break;
	}

	sum_ka += (T1[i] + T1[i + 9]) % 2;

}
cout << "9:" << sum_ka << endl;
sum_ka = 0;
for (int i = 0; i < T1.size(); i++)
{
	if (i == T1.size() - 10) {
		sum_ka += (T1[i] + T1[0]) % 2;
		sum_ka += (T1[i + 1] + T1[1]) % 2;
		sum_ka += (T1[i + 2] + T1[2]) % 2;
		sum_ka += (T1[i + 3] + T1[3]) % 2;
		sum_ka += (T1[i + 4] + T1[4]) % 2;
		sum_ka += (T1[i + 5] + T1[5]) % 2;
		sum_ka += (T1[i + 6] + T1[6]) % 2;
		sum_ka += (T1[i + 7] + T1[7]) % 2;
		sum_ka += (T1[i + 8] + T1[8]) % 2;
		sum_ka += (T1[i + 9] + T1[9]) % 2; break;
	}

	sum_ka += (T1[i] + T1[i + 10]) % 2;
```

```
	}
	cout << "10:" << sum_ka << endl;
}

int main() {
	setlocale(LC_ALL, "rus");
	vector <int> polinom_1{ 1,0,0,0,0,1,1,0,1,0,1,1,1,0,0,0,0,0,1,1,1 };
	vector <int> signal_1{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1 };
	vector <int> polinom_2{ 1,0,0,1,0,0,1,0,0,1,0,0,1,1,0,1,1,0,0,1,1,1,1 };
	vector <int> signal_2{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1 };

	//Заполняем
T1_____________________________________________________________________________________________
_____________________________________________
	vector <int> T1;
	int sum = 0;
	int new_chislo;
	while (signal_1 != polinom_1)
	{
		for (int i = 0; i < 21; i++)
		{
			sum += polinom_1[i] * signal_1[i];
		}
		new_chislo = sum % 2;
		signal_1.push_back(new_chislo);
		T1.push_back(signal_1[0]);
		signal_1.erase(signal_1.begin());
		sum = 0;
	}
	//Заполнили
T1_____________________________________________________________________________________________
_____________________________________________
	cout << "T для первого полинома:" << T1.size() << endl;

	//Заполняем
T2_____________________________________________________________________________________________
_____________________________________________
	vector <int> T2;
	sum = 0;
```

```
	while (signal_2 != polinom_2)
	{
		for (int i = 0; i < 21; i++)
		{
			sum += polinom_2[i] * signal_2[i];
		}
		new_chislo = sum % 2;
		signal_2.push_back(new_chislo);
		T2.push_back(signal_2[0]);
		signal_2.erase(signal_2.begin());
	}
	//Заполнили
T2_____________________________________________________________________________________________
_____________________________________________
	cout << "T для второго полинома:" << T2.size() << endl;
	polinoms(T1);
	polinoms(T2);
}
```

ВИСНОВОК

В ході практимума ми ознайомились з принципами побудови регістрів зсуву з лінійним зворотним зв’язком. А також практично освоїли їх програмну реалізацію. Та дослідили властивості лінійних рекурентних послідовностей та їх залежності від властивостей характеристичного полінома регістра.