**Порядок виконання роботи**  0. Уважно прочитати методичні вказівки до виконання комп’ютерного практикуму.

1. За даними характеристичними многочленами р1(x), р2(x) скласти лінійні рекурентні співвідношення для ЛРЗ, що задаються цими характеристичними многочленами.
2. Написати програми роботи кожного з ЛРЗ L1 , L2 .
3. За допомогою цих програм згенерувати імпульсні функції для кожного з ЛРЗ і підрахувати їх періоди.
4. За отриманими результатами зробити висновки щодо влавстивостей кожного з характеристичних многочленів р1(x), р2(x): многочлен примітивний над F2 ; не примітивний, але може бути незвідним; звідний.
5. Для кожної з двох імпульсних функцій обчислити розподіл k-грам на періоді, k≤nі, де nі - степінь полінома fі(x), і=1,2 а також значення функції автокореляції А(d) для 0≤ d ≤ 10. За результатами зробити висновки.

# P1(X)= X20+X17+X15+X14+X9+X7+X5+X3+X2+X+1

Period 1048575 примітивний Автокореляція:

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

# P2(X)= X24+ X22+ X18+ X17+ X16+ X15+ X12+ X11+ X9+ X4+ X2+ 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[]={1,1,1,0,1,0,0,0,0,1,0,1,1,0,0,1,1,1,1,0,0,0,1,0};//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[] = { 0,0,0,0,0,0,0,1,0,0,1,0,0,0,1,1,0,1,0,0,0,1,0,1,0,1,1,0,0,1,1,1,1,0,0,0,1,0,0,1,1,0,1,0,1,0,1,1,1,1,0,0,1,1,0,1,1,1,1,0,1,1,1,1 }; 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.