Национальный исследовательский университет «МЭИ» Институт автоматики и вычислительной техники

**Кафедра вычислительных машин, систем и сетей**

# Лабораторная работа № 6

# «Разработка программной реализации РСЛОС»

по курсу «Защита информации»

Выполнил: Кутузов И.Г.

Группа: А-08-16

Подпись: 

Преподаватель: Рытов А.А.

Москва, 2020 г.

In[ ]:=
```
(*Задание 1*)
(*Определить номер варианта nv=Nmod6+1 (N-номер по списку в группе),
                                                численное приближение
и выбрать многочлен для построения РСЛОС из таблицы:*)
Nomer = 10;
nv = Mod[Nomer, 6] + 1
     остаток от деления
polynom[x_] := 1 + x + x^3 x + x^4 + x^5
```

Out[ ]= 5

In[ ]:=
```
(*Задание 2*)
(*Составить таблицу соответствия степеней многочлена,
элементов РСЛОС и коэффициентов обратной связи.*)
```

| Многочлен | 1 * x^5 | 1 * x^4 | 1 * x^3 | 0 * x^2 | 1 * x^1 | 1 * x^0 |
|---|---|---|---|---|---|---|
| РСЛОС | | s4 | s3 | s2 | s1 | s0 |
| Коэффициенты обратной связи | | 1 | 1 | 0 | 1 | 1 |

;

In[ ]:=
```
(*Задние 4*)
(*Разработать программный модуль,реализующий работу заданного РСЛОС,
провести проверку работоспособности при начальной загрузке
 1Fh и получить выходную последовательность длиной 2*(25-1).*)
```

In[ ]:=
```
LSFR[in_, fb_, len_, out_] := Module[
                              программный модуль
  {init = FromDigits[in, 16], feedback = FromDigits[fb, 2], length = len, output = out},
          число по ряду цифр                 число по ряду цифр
  core = init;
  result = {};
  Do[
  оператор цикла
   oldcore = core;
   (*Задвинуть младший бит регистра в накопитель*)
   PrependTo[result, BitGet[core, 0]];
                     значение бита
   (*Вычислить результат сумматора*)
   fromXor = Mod[Total[IntegerDigits[BitAnd[core, feedback], 2]], 2];
             ос… сумм… цифры целого ч… побитный И
   (*Выдвинуть младший бит регистра и задвинуть результат сумматора*)
   core = BitShiftRight[core + BitShiftLeft[fromXor, StringLength[fb]]];
          сдвинуть биты вправо  сдвинуть биты влево  длина строки
   If[core == FromDigits["1e", 16] && output == True, Print[i]];
             число по ряду цифр                  истина печатать


   If[output == True, Print[ "Регистр до: ", IntegerString[oldcore, 2, StringLength[fb]],
   условный о… истина печатать                строковая запись целого числа  длина строки
     " | Результат сумматора: ", fromXor, " Выдвинутое значение: ", First[result],
                                                                    первый
     " Номер шага: ", i, " | Регистр после: ", IntegerString[core, 2,
                                                 строковая запись целого числа
      StringLength[fb]], " - " IntegerString[core, 16, StringLength[in]]]];
                               строковая запись целого числа длина строки
   , {i, length}];
  result
 ]

seq = LSFR["1f", "11011", 2 * (2^5 - 1), True];
                                         истина
```

```
Регистр до: 11111 | Результат сумматора: 0
  Выдвинутое значение: 1 Номер шага: 1 | Регистр после: 01111 - 0f
Регистр до: 01111 | Результат сумматора: 1
  Выдвинутое значение: 1 Номер шага: 2 | Регистр после: 10111 - 17
Регистр до: 10111 | Результат сумматора: 1
  Выдвинутое значение: 1 Номер шага: 3 | Регистр после: 11011 - 1b
Регистр до: 11011 | Результат сумматора: 0
  Выдвинутое значение: 1 Номер шага: 4 | Регистр после: 01101 - 0d
Регистр до: 01101 | Результат сумматора: 0
  Выдвинутое значение: 1 Номер шага: 5 | Регистр после: 00110 - 06
Регистр до: 00110 | Результат сумматора: 1
  Выдвинутое значение: 0 Номер шага: 6 | Регистр после: 10011 - 13
Регистр до: 10011 | Результат сумматора: 1
  Выдвинутое значение: 1 Номер шага: 7 | Регистр после: 11001 - 19
```

```
Регистр до: 11001 | Результат сумматора: 1
  Выдвинутое значение: 1 Номер шага: 8 | Регистр после: 11100 – 1c

Регистр до: 11100 | Результат сумматора: 0
  Выдвинутое значение: 0 Номер шага: 9 | Регистр после: 01110 – 0e

Регистр до: 01110 | Результат сумматора: 0
  Выдвинутое значение: 0 Номер шага: 10 | Регистр после: 00111 – 07

Регистр до: 00111 | Результат сумматора: 0
  Выдвинутое значение: 1 Номер шага: 11 | Регистр после: 00011 – 03

Регистр до: 00011 | Результат сумматора: 0
  Выдвинутое значение: 1 Номер шага: 12 | Регистр после: 00001 – 01

Регистр до: 00001 | Результат сумматора: 1
  Выдвинутое значение: 1 Номер шага: 13 | Регистр после: 10000 – 10

Регистр до: 10000 | Результат сумматора: 1
  Выдвинутое значение: 0 Номер шага: 14 | Регистр после: 11000 – 18

Регистр до: 11000 | Результат сумматора: 0
  Выдвинутое значение: 0 Номер шага: 15 | Регистр после: 01100 – 0c

Регистр до: 01100 | Результат сумматора: 1
  Выдвинутое значение: 0 Номер шага: 16 | Регистр после: 10110 – 16

Регистр до: 10110 | Результат сумматора: 0
  Выдвинутое значение: 0 Номер шага: 17 | Регистр после: 01011 – 0b

Регистр до: 01011 | Результат сумматора: 1
  Выдвинутое значение: 1 Номер шага: 18 | Регистр после: 10101 – 15

Регистр до: 10101 | Результат сумматора: 0
  Выдвинутое значение: 1 Номер шага: 19 | Регистр после: 01010 – 0a

Регистр до: 01010 | Результат сумматора: 0
  Выдвинутое значение: 0 Номер шага: 20 | Регистр после: 00101 – 05

Регистр до: 00101 | Результат сумматора: 1
  Выдвинутое значение: 1 Номер шага: 21 | Регистр после: 10010 – 12

Регистр до: 10010 | Результат сумматора: 0
  Выдвинутое значение: 0 Номер шага: 22 | Регистр после: 01001 – 09

Регистр до: 01001 | Результат сумматора: 0
  Выдвинутое значение: 1 Номер шага: 23 | Регистр после: 00100 – 04

Регистр до: 00100 | Результат сумматора: 0
  Выдвинутое значение: 0 Номер шага: 24 | Регистр после: 00010 – 02

Регистр до: 00010 | Результат сумматора: 1
  Выдвинутое значение: 0 Номер шага: 25 | Регистр после: 10001 – 11

Регистр до: 10001 | Результат сумматора: 0
  Выдвинутое значение: 1 Номер шага: 26 | Регистр после: 01000 – 08

Регистр до: 01000 | Результат сумматора: 1
  Выдвинутое значение: 0 Номер шага: 27 | Регистр после: 10100 – 14

Регистр до: 10100 | Результат сумматора: 1
  Выдвинутое значение: 0 Номер шага: 28 | Регистр после: 11010 – 1a

Регистр до: 11010 | Результат сумматора: 1
  Выдвинутое значение: 0 Номер шага: 29 | Регистр после: 11101 – 1d

30

Регистр до: 11101 | Результат сумматора: 1
  Выдвинутое значение: 1 Номер шага: 30 | Регистр после: 11110 – 1e
```

```
Регистр до: 11110 | Результат сумматора: 1
  Выдвинутое значение: 0 Номер шага: 31 | Регистр после: 11111 – 1f

Регистр до: 11111 | Результат сумматора: 0
  Выдвинутое значение: 1 Номер шага: 32 | Регистр после: 01111 – 0f

Регистр до: 01111 | Результат сумматора: 1
  Выдвинутое значение: 1 Номер шага: 33 | Регистр после: 10111 – 17

Регистр до: 10111 | Результат сумматора: 1
  Выдвинутое значение: 1 Номер шага: 34 | Регистр после: 11011 – 1b

Регистр до: 11011 | Результат сумматора: 0
  Выдвинутое значение: 1 Номер шага: 35 | Регистр после: 01101 – 0d

Регистр до: 01101 | Результат сумматора: 0
  Выдвинутое значение: 1 Номер шага: 36 | Регистр после: 00110 – 06

Регистр до: 00110 | Результат сумматора: 1
  Выдвинутое значение: 0 Номер шага: 37 | Регистр после: 10011 – 13

Регистр до: 10011 | Результат сумматора: 1
  Выдвинутое значение: 1 Номер шага: 38 | Регистр после: 11001 – 19

Регистр до: 11001 | Результат сумматора: 1
  Выдвинутое значение: 1 Номер шага: 39 | Регистр после: 11100 – 1c

Регистр до: 11100 | Результат сумматора: 0
  Выдвинутое значение: 0 Номер шага: 40 | Регистр после: 01110 – 0e

Регистр до: 01110 | Результат сумматора: 0
  Выдвинутое значение: 0 Номер шага: 41 | Регистр после: 00111 – 07

Регистр до: 00111 | Результат сумматора: 0
  Выдвинутое значение: 1 Номер шага: 42 | Регистр после: 00011 – 03

Регистр до: 00011 | Результат сумматора: 0
  Выдвинутое значение: 1 Номер шага: 43 | Регистр после: 00001 – 01

Регистр до: 00001 | Результат сумматора: 1
  Выдвинутое значение: 1 Номер шага: 44 | Регистр после: 10000 – 10

Регистр до: 10000 | Результат сумматора: 1
  Выдвинутое значение: 0 Номер шага: 45 | Регистр после: 11000 – 18

Регистр до: 11000 | Результат сумматора: 0
  Выдвинутое значение: 0 Номер шага: 46 | Регистр после: 01100 – 0c

Регистр до: 01100 | Результат сумматора: 1
  Выдвинутое значение: 0 Номер шага: 47 | Регистр после: 10110 – 16

Регистр до: 10110 | Результат сумматора: 0
  Выдвинутое значение: 0 Номер шага: 48 | Регистр после: 01011 – 0b

Регистр до: 01011 | Результат сумматора: 1
  Выдвинутое значение: 1 Номер шага: 49 | Регистр после: 10101 – 15

Регистр до: 10101 | Результат сумматора: 0
  Выдвинутое значение: 1 Номер шага: 50 | Регистр после: 01010 – 0a

Регистр до: 01010 | Результат сумматора: 0
  Выдвинутое значение: 0 Номер шага: 51 | Регистр после: 00101 – 05

Регистр до: 00101 | Результат сумматора: 1
  Выдвинутое значение: 1 Номер шага: 52 | Регистр после: 10010 – 12

Регистр до: 10010 | Результат сумматора: 0
  Выдвинутое значение: 0 Номер шага: 53 | Регистр после: 01001 – 09
```

```
Регистр до: 01001 | Результат сумматора: 0
  Выдвинутое значение: 1 Номер шага: 54 | Регистр после: 00100 – 04
Регистр до: 00100 | Результат сумматора: 0
  Выдвинутое значение: 0 Номер шага: 55 | Регистр после: 00010 – 02
Регистр до: 00010 | Результат сумматора: 1
  Выдвинутое значение: 0 Номер шага: 56 | Регистр после: 10001 – 11
Регистр до: 10001 | Результат сумматора: 0
  Выдвинутое значение: 1 Номер шага: 57 | Регистр после: 01000 – 08
Регистр до: 01000 | Результат сумматора: 1
  Выдвинутое значение: 0 Номер шага: 58 | Регистр после: 10100 – 14
Регистр до: 10100 | Результат сумматора: 1
  Выдвинутое значение: 0 Номер шага: 59 | Регистр после: 11010 – 1a
Регистр до: 11010 | Результат сумматора: 1
  Выдвинутое значение: 0 Номер шага: 60 | Регистр после: 11101 – 1d
61
Регистр до: 11101 | Результат сумматора: 1
  Выдвинутое значение: 1 Номер шага: 61 | Регистр после: 11110 – 1e
Регистр до: 11110 | Результат сумматора: 1
  Выдвинутое значение: 0 Номер шага: 62 | Регистр после: 11111 – 1f
```

In[ ]:=

```
Length[seq]
```

длина

```
seq
```

Out[ ]= 62

Out[ ]= {0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0,
1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1}

In[ ]:=

In[ ]:=
```
(*Задание 6*)
(*Сформировать программным путем матрицу m и вектор b,
с параметрами n=5 k=N+7,для решения системы линейных уравнений (m*c)mod2=b,
                    численное приближение
которая позволяет по 2n элементам последовательности
 S РСЛОС (см.п.4) определить коэффициенты обратной связи.*)
n = 5;
k = Nomer + 7;
k = 1;
S = Take[Reverse[seq], {k + 1, 2 * n + k}]
    изв··· расположить в обратном порядке
Length[S]
длина
m = {};
Do[AppendTo[m, Take[S, n]]; S = RotateLeft[S], n]
··· добавить в к··· извлечь          циклически сдвинуть влево
m // MatrixForm
     матричная форма
b = Take[S, n]
    извлечь
Reverse[LinearSolve[m, b, Modulus → 2]]
распол··· решить линейные ур··· модуль
```

Out[ ]= {1, 1, 1, 1, 0, 1, 1, 0, 0, 1}

Out[ ]= 10

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Out[ ]= {1, 1, 0, 0, 1}

Out[ ]= {1, 1, 0, 1, 1}

In[ ]:=
```
SeedRandom[Nomer]
инициализация генератора псевдослучайных чисел


pos = RandomInteger[Length[seq] - 2 * n];
      случайное цело··· длина
sample = Take[Reverse[seq], {pos + 1, pos + 2 * n}]
         изв··· расположить в обратном порядке
Length[sample]
длина
s = sample
```

Out[ ]= {1, 1, 0, 0, 0, 0, 1, 1, 0, 1}

Out[ ]= 10

Out[ ]= {1, 1, 0, 0, 0, 0, 1, 1, 0, 1}

```
In[ ]:= BM[sample_] := Module[{s = sample},
                      программный модуль
  Lol = 0; fol = 1;
 diff = 0; Clear[x];
           очистить
 f = 1;
  L = 0;
  g = CoefficientList[f, {x}];
      список коэффициентов многочлена
  Do[If[Mod[∑_{i=1}^{L} g[[i]] s[[j - 1 - L + i]], 2] == s[[j]], diff = diff + 1,
  ... y... остаток от деления
    Lne = Max[j - L, L];          fne = PolynomialMod[x^(Lne-L) f + x^(Lne-Lol-diff-1) fol, 2];
          максимум                      упростить многочлен по модулю
    If[Lne ≠ L, fol = f;
    условный оператор
     Lol = L;
     L = Lne;
     diff = 0, diff = diff + 1];
    f = fne;
    g = CoefficientList[f, {x}]];
        список коэффициентов многочлена
   Print["j=", j, ", L=", L, ", f=", f], {j, Length[s]}]
   печатать                              длина
 ]
```

```
In[ ]:= BM[sample]
```

j=1, L=1, f=1 + x

j=2, L=1, f=1 + x

j=3, L=2, f=1 + x + $x^2$

j=4, L=2, f=$x^2$

j=5, L=2, f=$x^2$

j=6, L=2, f=$x^2$

j=7, L=5, f=1 + x + $x^5$

j=8, L=5, f=1 + x + $x^4$ + $x^5$

j=9, L=5, f=1 + x + $x^3$ + $x^4$ + $x^5$

j=10, L=5, f=1 + x + $x^3$ + $x^4$ + $x^5$

```
In[ ]:= (*Многочлен совпадает с исходным*)
```

In[ ]:=
```
(*Задание 9*)
(*На базе трех РСЛОС с номерами вариантов nv,(nv+2)mod6,
(nv+4)mod6,сформировать программную реализацию генератора
 Геффе.Использовать поразрядные логические операторы:BitAnd[],
                                                    побитный И
BitNot[],BitXor[].Получить последовательность в 150 бит*)
          сложение битов по модулю 2
```

In[ ]:=
```
Mod[nv + 2, 6]
остаток от делени
Mod[nv + 4, 6]
остаток от делени
```

Out[ ]= 1

Out[ ]= 3

In[ ]:=
```
lsfr1 = FromDigits[LSFR["A", "11011", 150, False], 2]
        число по ряду цифр                  ложь
lsfr2 = FromDigits[LSFR["C", "00101", 150, False], 2]
        число по ряду цифр  генерируемая конст··· ложь
lsfr3 = FromDigits[LSFR["E", "01111", 150, False], 2]
        число по ряду цифр  основание натурал··· ложь
geffe = BitXor[BitAnd[lsfr1, lsfr2], BitAnd[BitNot[lsfr1], lsfr3]]
        сложе··· побитный И          побит··· битное отрицание
geffelist = IntegerDigits[geffe, 2, 150]
            цифры целого числа
```

Out[ ]= 643 870 273 449 631 342 179 984 310 705 328 487 387 755 594

Out[ ]= 860 031 961 431 650 147 564 892 575 135 016 528 084 798 188

Out[ ]= 582 018 581 158 003 873 053 334 214 590 029 277 530 526 702

Out[ ]= 146 370 001 792 022 319 314 572 867 057 857 310 319 248 364

Out[ ]=
```
{0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1,
 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0,
 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1,
 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0}
```

In[ ]:=
```
len = Max[BitLength[lsfr1], BitLength[lsfr2], BitLength[lsfr3]]
      ма·· длина в битах     длина в битах     длина в битах
```

Out[ ]= 150

In[ ]:=
```
(*Задание 10*)
(*Определить линейную сложность генератора Геффе.*)
(BitLength[lsfr1] + 1) * BitLength[lsfr2] + BitLength[lsfr1] * BitLength[lsfr3]
 длина в битах          длина в битах          длина в битах          длина в битах
```

Out[ ]= 44 701