Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**Лабораторная работа №2**

по дисциплине: «Информационная безопасность и защита информации».

Выполнил:

студент 4 курса, гр. ИВТАПбд-41

Кондратьев Павел Сергеевич.

Проверил:

преподаватель кафедры ВТ

Мартынов Антон Иванович.

г. Ульяновск, 2019

**1) Задание:**

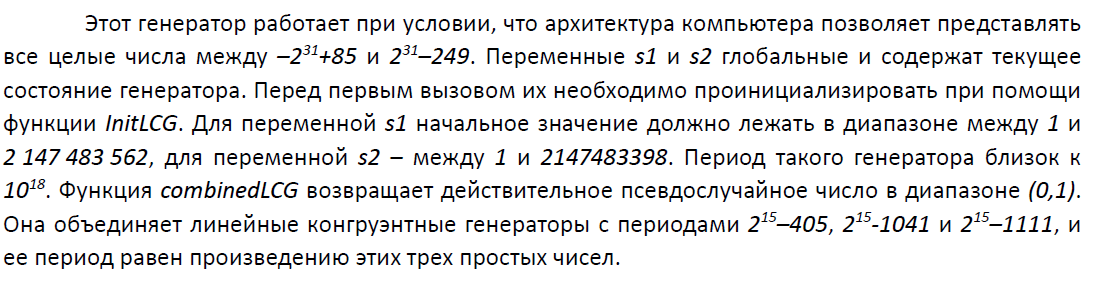
Реализовать приложение интерфейсом, позволяющее выполнять следующие действия:

1. Генерировать последовательность в битах (длина последовательности задается) с
2. помощью выбранного генератора
3. Сохранять полученную последовательность в файл и выводить ее на экран приложения
4. Загружать последовательность из файла
5. Проверять полученную последовательность с помощью тестов, реализованных в предыдущей лабораторной работе. Результат проверки должен отображаться в приложении

**2) Краткие теоретические сведения:**

Кубический конгруэнтный генератор задается как

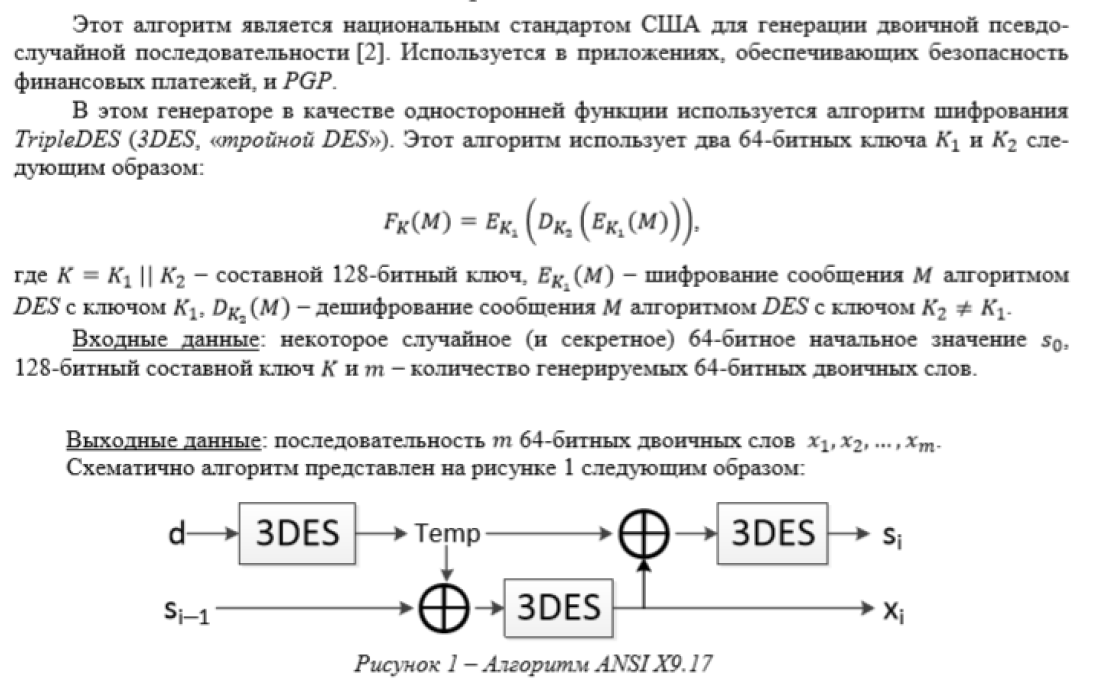


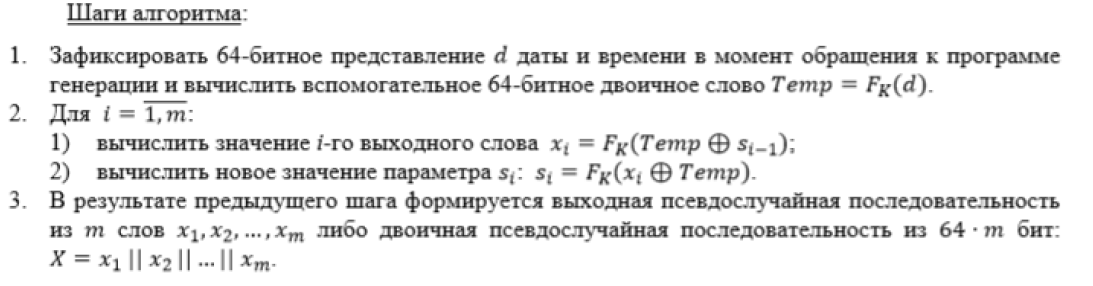


Одним из комбинирующих генераторов является генератор Геффа. В нем используются три LFSR, объединенные нелинейным способом. LFSR-2 и LFSR-3 являются входами мультиплексора (рабочие регистры), а третий управляет входом мультиплексора. Если длины LFSR равны n1, n2, n3 соответственно, то линейная сложность генератора равна (n1 + 1) \* n2 + n1 \* n3. Период такого генератора будет равен наименьшему общему делителю периодов трех генераторов. При условии, что размеры регистров взаимно просты, то период этого генератора будет равен произведению периодов трех LFSR. В обобщенной схеме генератора Геффа используются несколько рабочих LFSR.



**Алгоритм ANSI X9.17**





**3) Порядок выполнения работы:**

1. Программа должна быть оформлена в виде удобной утилиты с интерактивным интерфейсом пользователя
2. Текст программы оформляется прилично (удобочитаемо, с описанием ВСЕХ функций, переменных и критических мест).
3. В процессе работы программа ОБЯЗАТЕЛЬНО выдает информацию о состоянии процесса генерации / тестирования (если процесс занимает длительное время)
4. Интерфейс программы может быть произвольным, но удобным и понятным (разрешается использование библиотек GUI)
5. Среда разработки и язык программирования могут быть произвольными



Рис1. Форма для задания длинны последовательности



Рис2. Страница выбора файла с генерированной последовательностью

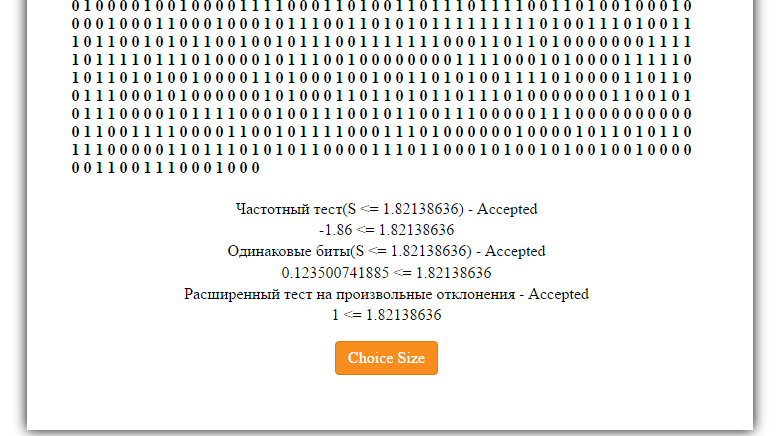


Рис3. Результат программы

**Вывод:**

В ходе выполнения данной лабораторной работы, мне удалось освоить базовую разработку графического интерфейса. Я научился тестировать последовательность бит на равномерность и случайность и научиться генерировать псевдослучайную последовательность с помощью различных алгоритмов. Также рассмотрел 3 алгоритма проверки на последовательность бит (Расширенный тест на произвольные отклонения Тест на последовательность одинаковых бит Частотный тест) и 3 алгоритма генерации псевдо последовательных чисел (Кубический конгруэнтный генератор, Генератор Геффа, ANSI X9.17).

**Список литературы:**

* Руководство по PHP https://www.php.net/manual/ru/index.php (Дата обращения 2.12.19).
* Лабораторная работа Мартынов Антон Иванович - Информационная безопасность и защита информации.
* Криптографические генераторы. Поточные шифры и их криптоанализ http://cryptowiki.net/index.php? (Дата обращения 8.12.19).
* Генераторы случайных и Псевдослучайных последовательностей. Статистические тесты. Криптографически безопасные генераторы псевдослучайных последовательностей. [https://www.nrjetix.com/ fileadmin/doc/publications/Lectures\_security/Lecture2.pdf](https://www.nrjetix.com/%20fileadmin/doc/publications/Lectures_security/Lecture2.pdf) (Дата обращения 8.12.19).

**Приложение 1**

function congruent\_cube\_generator($n) {

// n - число записей в выходном файле, длина последовательности, которую создает генератор

$sequence = "";

$a0 = 0; //первое число в генерируемой последовательности

$a = 106; $b = 1283; $c = 7; $d = 5; // 3 3 7 5

$m = 6075; //модуль по которому будем приводить | 2

$a\_prev = $a0;

for($count = 1; $count < $n + 1; $count++) {

$a\_next = $a \* pow($a\_prev, 3);

$a\_next = $a\_next + $b \* pow($a\_prev, 2);

$a\_next = $a\_next + $c \* $a\_prev;

$a\_next = $a\_next + $d;

$a\_next = ($a\_next % $m); //приведение результата по модулю

$a\_prev = $a\_next;

$sequence .= $a\_next;

$sequence .= ' ';

}

return $sequence;

}

function Geffe($L1, $L2, $L3, $n) {

$holder = "";

for($i = 0; $i < $n; $i++) {

$L1 = ($L1 << 1) | ((($L1 >> 29)^ ($L1 >> 28) ^ ($L1 >> 25) ^ ($L1 >> 23)) & 1);

$L2 = ($L2 << 1) | ((($L2 >> 30)^ ($L2 >> 27)) & 1);

$L3 = ($L3 << 1) | ((($L3 >> 31)^ ($L3 >> 30) ^ ($L3 >> 29) ^ ($L3 >> 28) ^ ($L3 >> 26) ^ ($L3 >> 24)) & 1 );

$holder .= (((($L3 >> 32) & 1 )\*(($L1 >> 30) & 1)) ^ (((($L3 >> 32) & 1) ^ 1) \* (($L2 >> 31) & 1)) );

$holder .= ' ';

}

return $holder;

}

function \_Make64 ( $hi, $lo ) {

// on x64, we can just use int

if ( ((int)4294967296)!=0 )

return (((int)$hi)<<32) + ((int)$lo);

// workaround signed/unsigned braindamage on x32

$hi = sprintf ( "%u", $hi );

$lo = sprintf ( "%u", $lo );

// use GMP or bcmath if possible

if ( function\_exists("gmp\_mul") )

return gmp\_strval ( gmp\_add ( gmp\_mul ( $hi, "4294967296" ), $lo ) );

if ( function\_exists("bcmul") )

return bcadd ( bcmul ( $hi, "4294967296" ), $lo );

// compute everything manually

$a = substr ( $hi, 0, -5 );

$b = substr ( $hi, -5 );

$ac = $a\*42949; // hope that float precision is enough

$bd = $b\*67296;

$adbc = $a\*67296+$b\*42949;

$r4 = substr ( $bd, -5 ) + + substr ( $lo, -5 );

$r3 = substr ( $bd, 0, -5 ) + substr ( $adbc, -5 ) + substr ( $lo, 0, -5 );

$r2 = substr ( $adbc, 0, -5 ) + substr ( $ac, -5 );

$r1 = substr ( $ac, 0, -5 );

while ( $r4>100000 ) { $r4-=100000; $r3++; }

while ( $r3>100000 ) { $r3-=100000; $r2++; }

while ( $r2>100000 ) { $r2-=100000; $r1++; }

$r = sprintf ( "%d%05d%05d%05d", $r1, $r2, $r3, $r4 );

$l = strlen($r);

$i = 0;

while ( $r[$i]=="0" && $i<$l-1 )

$i++;

return substr ( $r, $i );

}

function ANSIX($n) {

$holder = "";

$s = round(rand(0, 1));

for($i = 0; $i < $n; $i++) {

$I = crypt(strftime("%H").date("isdmY"), 'rl');

$q = \_Make64($I,round(rand(0, 1)));

$x[$i] = $q^$s;

$s = $x[$i]^$q;

$holder .= $x[$i];

$holder .= ' ';

}

echo $holder;

return $holder;

}

// $sequence = congruent\_cube\_generator($size);

// $l1 = round(rand(1, 29)); // 29

// $l2 = round(rand(1, 30)); // 30

// $l3 = round(rand(1, 31)); // 31

// $sequence = Geffe($l1, $l2, $l1, $size);

// $sequence = ANSIX($size);