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

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**Лабораторная работа №1**

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, позволяющее выполнять следующие действия:

1. Задавать длину генерируемой последовательность в битах (при тестировании рекомендуется задавать длину последовательности не менее 10 000 бит)

2. Генерировать псевдослучайную последовательность 0 и 1 с помощью стандартного алгоритма генерации случайных чисел

3. Загружать последовательность из текстового файла

4. Сохранять полученную последовательность в файл и выводить ее на экран приложения

5. Проверять полученную последовательность с помощью реализованных тестов. Результат проверки должен отображаться в приложении

**2) Краткие теоретические сведения:**

**Частотный тест**

Этот тест оценивает пропорцию нулей и единиц в проверяемой последовательности. Тест определяет, является ли количество нулей и единиц в последовательности приблизительно таким же, как должно быть в истинно случайной последовательности.

**Шаги алгоритма:**

1. Входная последовательность, состоящая из 0 и 1 (будем обозначать ее ε), преобразовывается в последовательность -1 и 1 (будем обозначать ее X) соответственно:

𝑋𝑖=2∗𝜀𝑖−1

2. Вычисляется сумма 𝑆𝑛=𝑋1+𝑋2+ 𝑋3+⋯+ 𝑋𝑛, где n – количество элементов проверяемой последовательности.

3. Вычисляется статистика 𝑆=|𝑆𝑛|√𝑛

4. Если 𝑆≤1.82138636, то тест считается успешно пройденным, иначе делается вывод о том, что последовательность является неслучайной.

**Тест на последовательность одинаковых бит**

Этот тест анализирует количество цепочек в проверяемой последовательности, где цепочка – это непрерывная последовательность одинаковых бит. Под цепочкой длиной k понимается цепочка, состоящая из ровно k бит и ограниченная до и после битами с противоположным значением. Тест определяет, является ли количество цепочек из нулей и единиц различной длины в последовательности приблизительно таким же, как должно быть в истинно случайной последовательности

**Шаги алгоритма:**

1. Вычисляется частота, с которой в проверяемой последовательности встречаются единицы:

𝜋=1𝑛∗Σ𝜀𝑗𝑛𝑗=1

2. Вычисляется значение 𝑉𝑛=1+Σ𝑟(𝑘)𝑛−1𝑘=1, где r(k) = 0, если 𝜀𝑘=𝜀𝑘+1 и r(k)=1 иначе

3. Вычисляется статистика 𝑆=|𝑉𝑛−2∗𝑛∗𝜋∗(1−𝜋)|2∗√2∗𝑛∗𝜋∗(1−𝜋)

4. Если 𝑆≤1.82138636, то тест считается успешно пройденным, иначе делается вывод о том, что последовательность является неслучайной.

**Расширенный тест на произвольные отклонения**

Этот тест оценивает общее число посещений определенного состояния при произвольном обходе кумулятивной суммы. Цель этого теста – определить отклонения от ожидаемого числа посещений различных состояний при произвольном обходе. Фактически данный тест состоит из 18 тестов, по одному для каждого состояния: −9,−8,…−1,1,2,…,9.

**Шаги алгоритма:**

1. Входная последовательность, состоящая из 0 и 1 (будем обозначать ее ε), преобразовывается в последовательность -1 и 1 (будем обозначать ее X) соответственно:

𝑋𝑖=2∗𝜀𝑖−1

2. Вычисляются суммы 𝑆𝑖 последовательно удлиняющихся подпоследовательностей, начинающихся с 𝑋1

𝑆1= 𝑋1 𝑆2=𝑋1+ 𝑋2 𝑆3=𝑋1+ 𝑋2+𝑋3 … 𝑆𝑛=𝑋1+ 𝑋2+𝑋3+⋯+𝑋𝑛

3. Формируется новая последовательность 𝑆′=0,𝑆1,𝑆2,…,𝑆𝑛,0

4. Вычисляется 𝐿=𝑘−1, где k – количество нулей в полученной последовательности 𝑆′.

5. Для каждого из 18 состояний вычисляется 𝜉𝑗, которое показывает, сколько раз состояние j встречалось в последовательности 𝑆′. Здесь j = −9,−8,…−1,1,2,…,9.

6. Вычисляются 18 статистик 𝑌𝑗=|𝜉𝑗−𝐿|√2∗𝐿∗(4∗|𝑗|−2) для каждого состояния j = −9,−8,…−1,1,2,…,9.

7. Если все статистики 𝑌𝑗≤1.82138636, то тест считается успешно пройденным, если же хотя бы для одной статистики 𝑌𝑗 это условие не выполнилось, то делается вывод о том, что последовательность является неслучайной.

**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

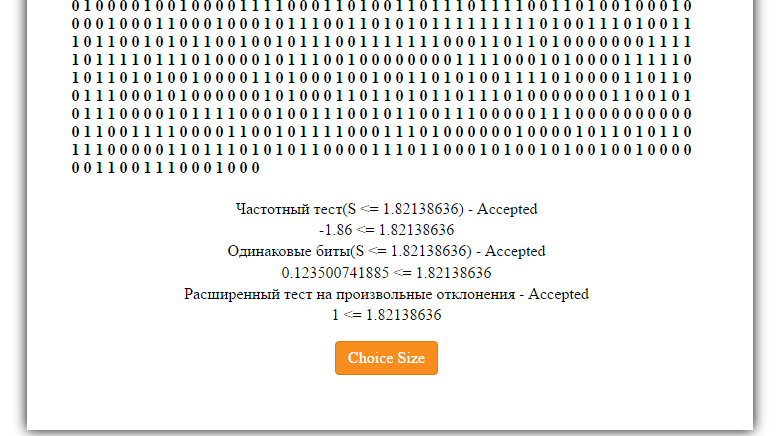


Рис3. Результат программы

**Вывод:**

В ходе выполнения данной лабораторной работы, мне удалось освоить базовую разработку графического интерфейса. Я научился тестировать последовательность бит на равномерность и случайность. Также рассмотрел 3 алгоритма проверки на последовательность бит (Расширенный тест на произвольные отклонения Тест на последовательность одинаковых бит Частотный тест).

**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.

**Приложение 1**

<?php

if($\_POST['command-name'] == "debug") {

$size = $\_POST['Size'];

$sequence = "";

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

$sequence .= round(rand(0, 1));

$sequence .= ' ';

}

// //делается попытка создать его

$fp = fopen("file.txt", "w");

// записываем в файл текст

fwrite($fp, $sequence);

// закрываем

fclose($fp);

}

function test\_1($data, $len) {

$s = 0;

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

$s += 2 \* $data[$i] - 1;

}

$result = $s / sqrt($len);

if($result <= 1.82138636) {

echo "<p>Частотный тест(S <= 1.82138636) - Accepted</p><p>$result <= 1.82138636</p>";

} else {

echo "<p>Частотный тест(S <= 1.82138636) - Falled</p><p>$result <= 1.82138636</p>";

}

}

function r($first, $second) {

if($first === $second) {

return 0;

} else {

return 1;

}

}

function test\_2($data, $len) {

$p = array\_sum($data) / $len;

$v = 1;

for($i = 1; $i < $len - 1; $i++) {

$v += r($data[$i], $data[$i + 1]);

}

$result = abs($v - 2 \* $len \* $p \* (1 - $p)) / (2 \* sqrt(2 \* $len) \* $p \* (1 - $p));

if($result <= 1.82138636) {

echo "<p>Одинаковые биты(S <= 1.82138636) - Accepted</p><p>$result <= 1.82138636</p>";

} else {

echo "<p>Одинаковые биты(S <= 1.82138636) - Falled</p><p>$result <= 1.82138636</p>";

}

}

function test\_3($data, $len) {

$s = array();

$zero = $tmp = 0;

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

$s[$i] += $tmp + 2 \* $data[$i] - 1;

$tmp = $s[$i];

if($s[$i] == 0) {

$zero++;

}

}

$new\_s = $s; array\_unshift($new\_s, 0); array\_push($new\_s, 0);$new\_len = count($new\_s);

for($i = 0; $i < $new\_len; $i++) {

if($new\_s[$i] == 0)

$zero++;

}

$l = $zero - 1;

$j = array(

-9 => 0,-8 => 0,-7 => 0,-6 => 0,-5 => 0,-4 => 0,-3 => 0,-2 => 0,-1 => 0,

0 => 0, 1 => 0, 2 => 0, 3 => 0, 4 => 0, 5 => 0, 6 => 0, 7 => 0, 8 => 0, 9 => 0

);

for($i = -9; $i <= 9; $i++) {

for($h = 0; $h < $new\_len; $h++) {

if($new\_s[$h] == $i)

$j[$i]++;

}

}

$y = array(

-9 => 0,-8 => 0,-7 => 0,-6 => 0,-5 => 0,-4 => 0,-3 => 0,-2 => 0,-1 => 0,

0 => 0, 1 => 0, 2 => 0, 3 => 0, 4 => 0, 5 => 0, 6 => 0, 7 => 0, 8 => 0, 9 => 0

);

for($i = -9; $i <= 9; $i++) {

$y[$i] = (abs($j[$i] - $l)) / sqrt(2 \* $l \* (4 \* abs($j) - 2));

if($y[$i] <= 1.82138636) {

$result = true;

} else {

$result = false;break;

}

}

if($result) {

echo "<p>Расширенный тест на произвольные отклонения - Accepted</p><p>$result <= 1.82138636</p>";

} else {

echo "<p>Расширенный тест на произвольные отклонения - Falled</p><p>$result <= 1.82138636</p>";

}

}

?>