Федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский политехнический университет Петра Великого»

Неделя науки 2019

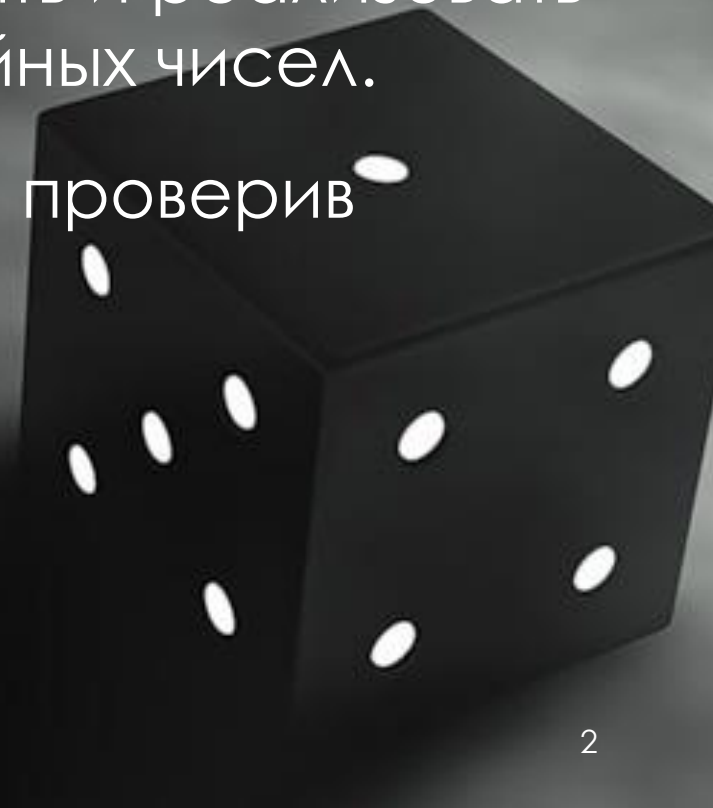
Методы генерации случайных чисел

Выполнили:
Неверов Вадим (гр. 3530903/80002)
Кюн Анастасия (гр. 3530203/80002)

Научный руководитель: доцент каф. Высшая математика
Филимоненкова Надежда Викторовна

Задачи проекта

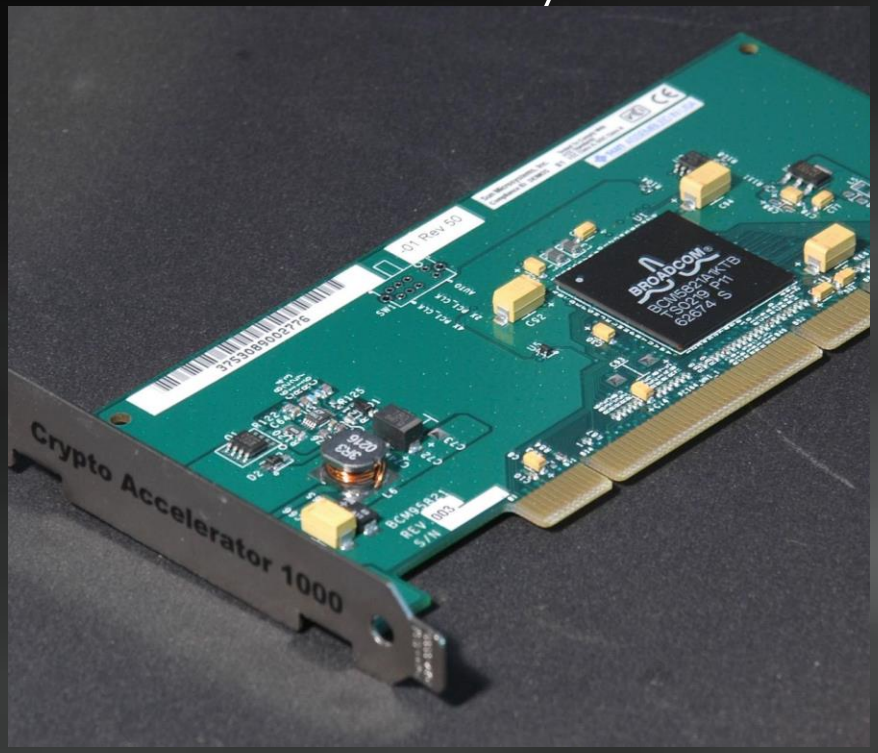
1. Изучить критерии случайности и методы генерации случайных чисел.
2. Самостоятельно разработать и реализовать алгоритм генерации случайных чисел.
3. Оценить работу алгоритма, проверив случайность генерируемой последовательности чисел.



Сферы применения случайных и псевдослучайных чисел:

- наука
- криптография
- игры
- искусство
- политика

Аппаратный генератор случайных чисел



Аппаратный генератор случайных чисел – устройство, которое генерирует последовательность случайных чисел на основе хаотически изменяющихся параметров физических процессов.

Псевдослучайные числа

Свойства псевдослучайных чисел близки к свойствам случайных. Отличие псевдослучайного числа от случайного состоит в том, что оно получено путём вычислений по некоторому алгоритму, как следствие, для генераторов псевдослучайных чисел возможно предсказать, какое число будет сгенерировано.

Генератор Кнута

K1. [Выбрать число итераций.] Присвоить Y наибольшую значащую цифру X.

K2. [Выбрать случайный шаг] Присвоить следующую наибольшую значащую цифру X. Переходим к шагу K(3 + Z), т. е. к случайно выбранному шагу в программе.

K3. [Обеспечить > 5 x 109] Если X < 5000000000, присвоить X значение X + 5000000000.

K4. [Средина квадрата.] Заменить X серединой квадрата X.

…

K13. [Повторить?] Если Y > 0, уменьшить У на 1 и возвратиться к шагу K2. Если Y = 0, алгоритм завершен. Значение числа X, полученное на предыдущем шаге, и будет желаемым «случайным» значением.

6065038420

Линейный конгруэнтный метод

$$X_{n+1} = (aX_n + \mathrm{c}) \bmod \mathrm{m}$$

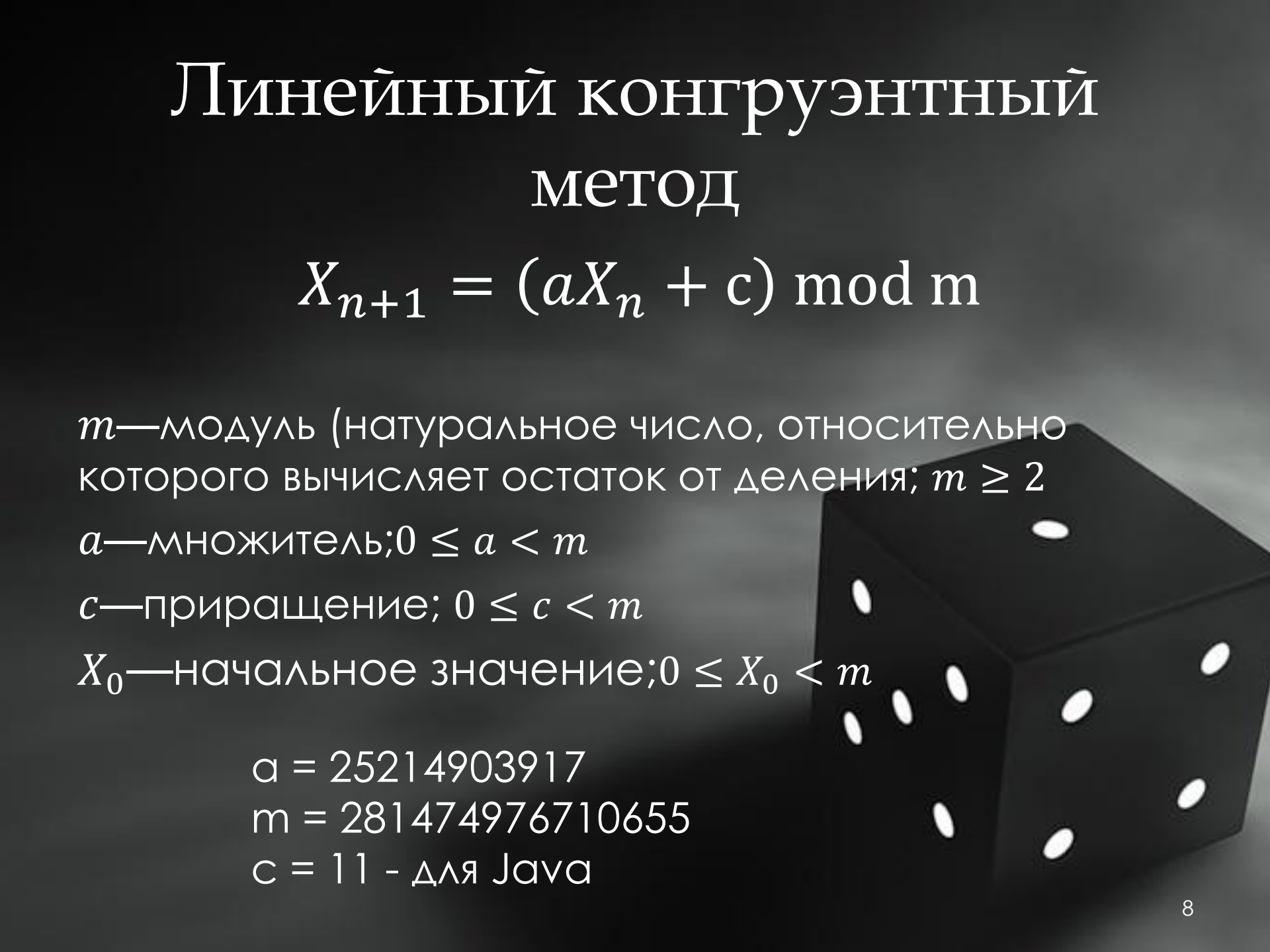
m—модуль (натуральное число, относительно которого вычисляет остаток от деления; $m \geq 2$

a—множитель;$0 \leq a < m$

c—приращение; $0 \leq c < m$

X_0—начальное значение;$0 \leq X_0 < m$

a = 25214903917
m = 281474976710655
c = 11 - для Java



Еще примеры алгоритмов

- RANDU
- Вичмана-Хилла
- Генератор Макларена — Марсальи
- Вихрь Мерсенна

Разработка и реализация алгоритма генерации случайных чисел

На чем не может быть основан алгоритм:

- Музыка
- Фото
- Текст
- Действия и решения одного человека

Разработка и реализация алгоритма генерации случайных чисел

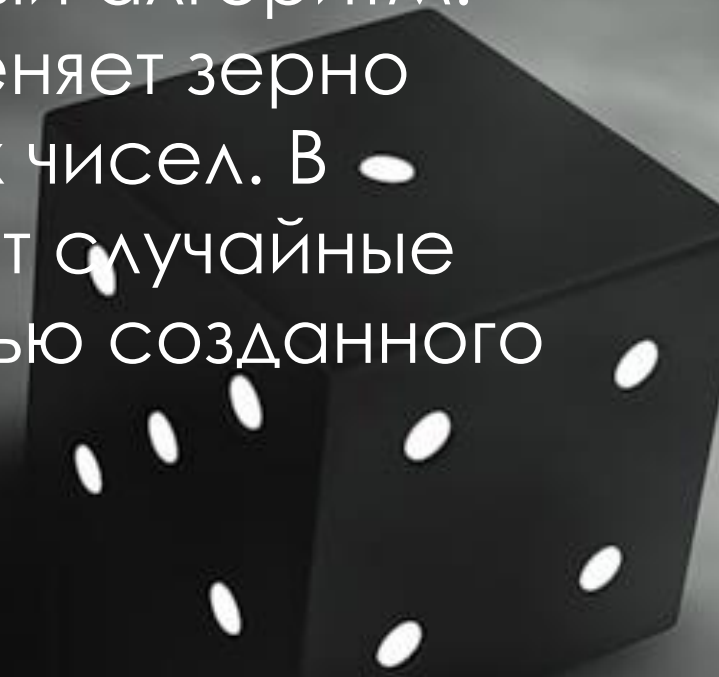
Гадалка Шеннона

5-индексный массив A[] из 72 элементов

Если человек последними написал числа a_1, a_2, a_3 и машина на это отвечала b_1, b_2, b_3, то в ячейку A[a_1, b_1, a_2, b_2, a_3] добавляется единица, то есть машина запоминает, что после комбинации a_1, b_1, a_2, b_2 человек выбрал число a_3. Чтобы предсказать, что теперь напишет человек, машина сравнивает числа A[a_2, b_2, a_3, b_3, 0] и A[a_2, b_2, a_3, b_3, 1].

Разработка и реализация алгоритма генерации случайных чисел

Нами был создан смешанный алгоритм. Это алгоритм Alea, но он меняет зерно каждые 10 сгенерированных чисел. В качестве зерна же выступают случайные числа, собранные с помощью созданного нами скрипта



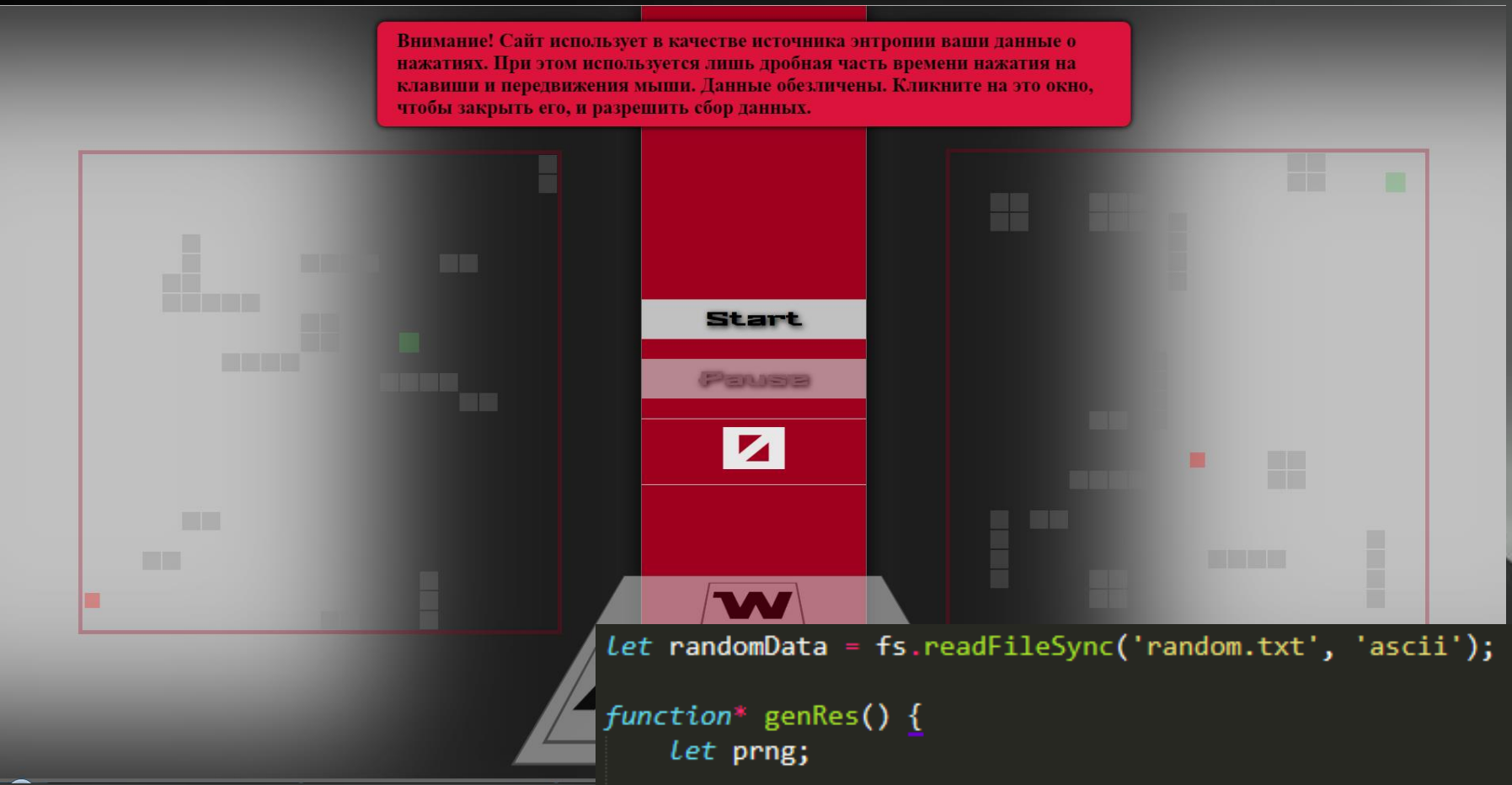
```javascript
let settings = {
  measurementDelay: 9,
  saveInterval: 3500,
  warningMessage: "Внимание! Сайт использует в качестве источника энтропии ваши данные о
}
//First show warning. When warning is clicked, data begins to flow.
showWarning();

let randomData = "";

document.addEventListener('mousemove', debounce(function(e
  randomData += e.timeStamp.toString().slice(-2);
}, settings.measurementDelay));

document.addEventListener('keyup', debounce(function(e) {
  randomData += e.timeStamp.toString().slice(-2);
}, settings.measurementDelay));

document.addEventListener('mouseup', debounce(function(e)
  randomData += e.timeStamp.toString().slice(-2);
}, settings.measurementDelay));
```

Editor: save.php

Файл ▾ Кодировка ▾ Синтаксис ▾ Поиск

```php
<?php
header('Access-Control-Allow-Origin: *');

$curNum = file_get_contents('currentNumber.txt');

$data = $_GET['randomData'];
$file = fopen("randomData$curNum.txt", 'a');
fwrite($file, $data);
fclose($file);

if (filesize("randomData$curNum.txt") > 50000) {

    fclose(fopen("currentNumber.txt",'w'));

    $currentNumNew = fopen("currentNumber.txt", 'w');
    fwrite($currentNumNew, $curNum+1);
    fclose($currentNumNew);

    echo "New file created";
} else {
    echo filesize("randomData$curNum.txt");
}
```

Режим: php | utf-8 | Loaded 531 bytes




```
let randomData = fs.readFileSync('random.txt', 'ascii');

function* genRes() {
    let prng;

    for (let i = 0;i<Infinity;i+=10) {
        prng = Alea(randomData.slice(i, i+10));

        for (let j = 0;j<25;j++) {
            yield prng().toString().split(".")[1].slice(0,12);
        }
    }
}
```

Как проверить случайность последовательности?

- Спектральный тест
- Тест рангов бинарных матриц
- Побитовый частотный тест

Особенности тестирования

- P-величина – вероятность воспроизведения идеальным генератором последовательности хуже, чем исследуемая.


NIST
National Institute of
Standards and Technology
U.S. Department of Commerce
node

Частотный побитовый тест

$$\varepsilon = 1011010101$$

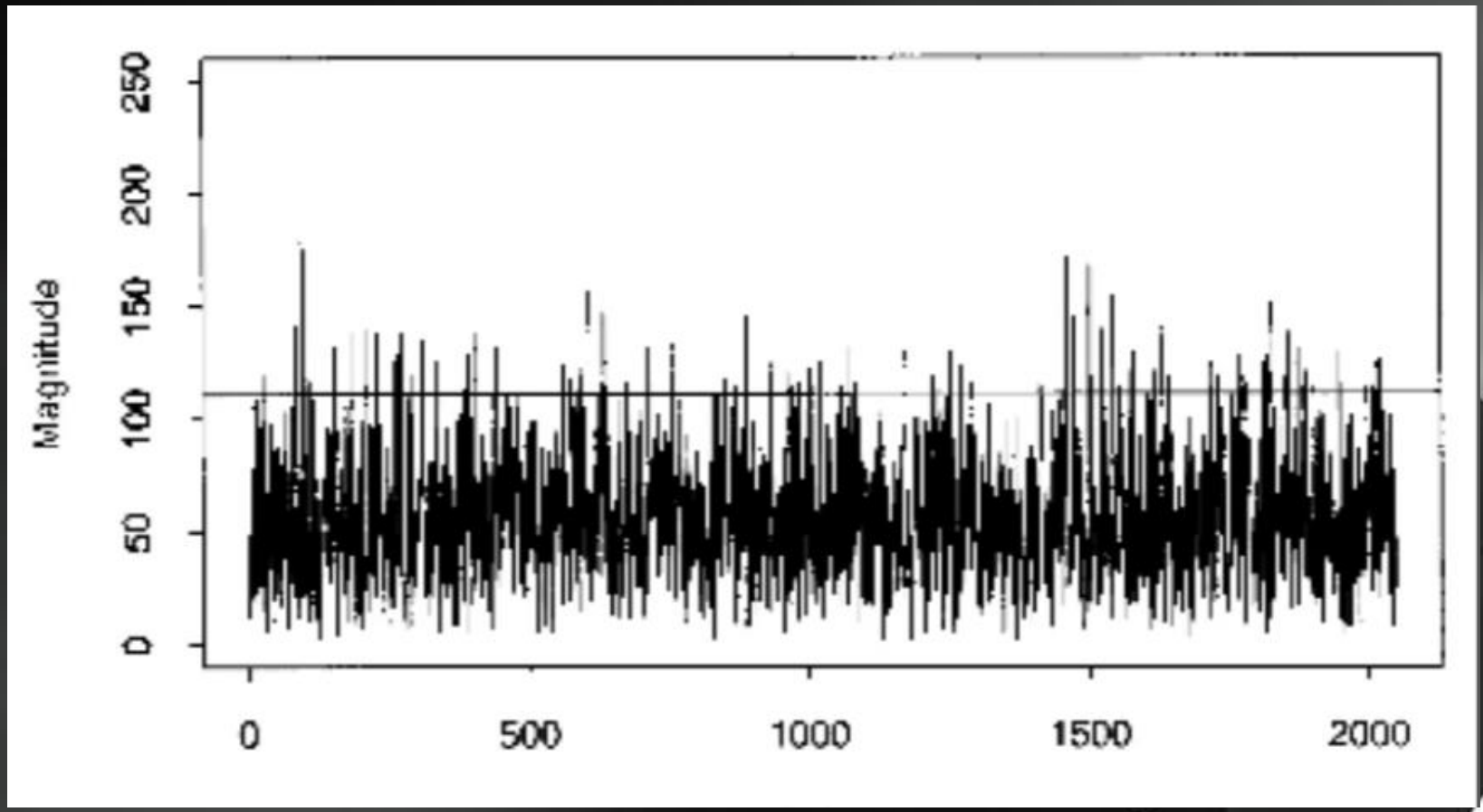
$$S_n = 1 + (-1) + 1 + 1 + (-1) + 1 + (-1) + 1 + (-1) + 1 = 2$$

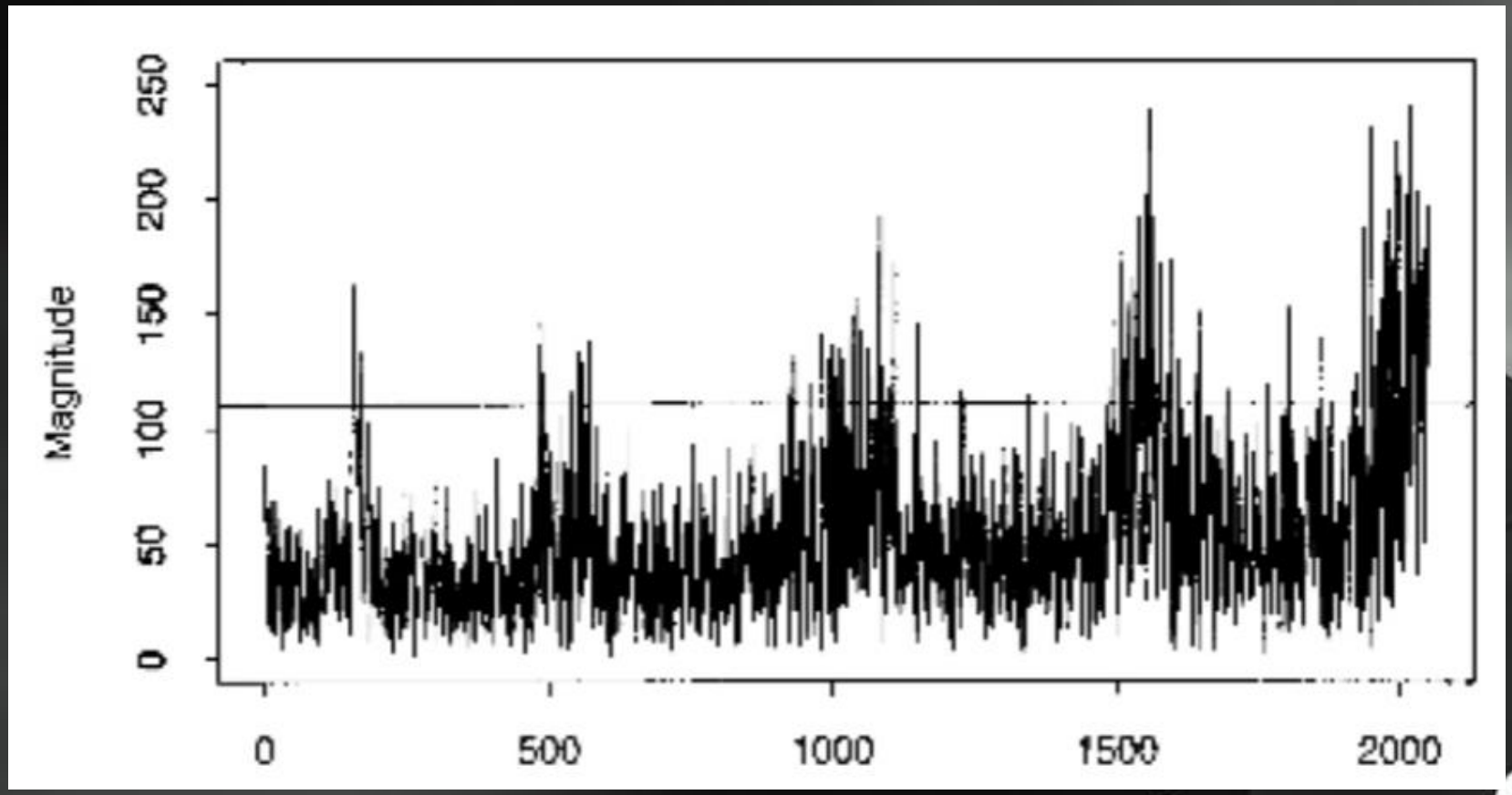
$$S_{obs} = \frac{|S_n|}{\sqrt{n}}$$

$$P - value = \text{erfc}(\frac{632455532}{\sqrt{2}}) = 0.527089$$

Спектральный тест

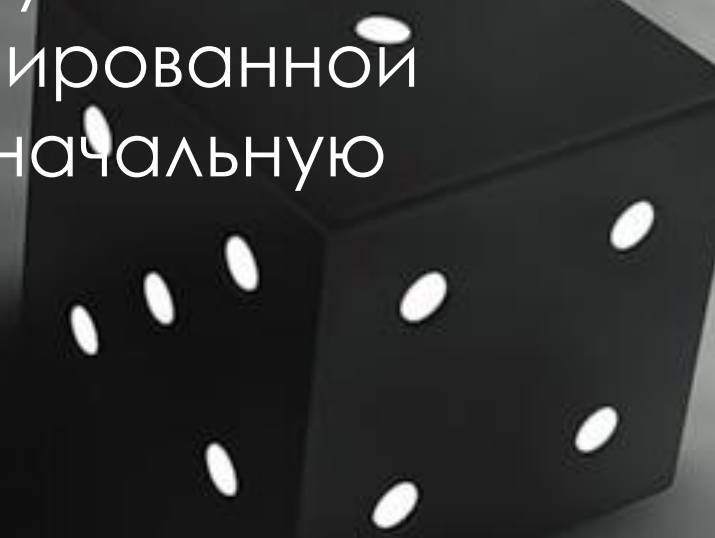
Суть теста заключается в оценке высоты пиков дискретного преобразования Фурье исходной последовательности. Цель — выявление периодических свойств входной последовательности, например, близко расположенных друг к другу повторяющихся участков.


Magnitude
0
50
100
150
200
250
0
500
1000
1500
2000


Magnitude
0
50
100
150
200
250
0
500
1000
1500
2000

Тест рангов бинарных матриц

Здесь производится расчёт рангов непересекающихся подматриц, построенных из исходной двоичной последовательности. Целью этого теста является проверка на линейную зависимость подстрок фиксированной длины, составляющих первоначальную последовательность.



Пусть M = Q = 3 (M и Q – количества строк и столбцов каждой матрицы)

01011001001010101101

$$\begin{vmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{vmatrix} \qquad \begin{vmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{vmatrix}$$

F_M – количество матриц с рангом M

F_{M-1} – количество матриц с рангом M-1

N– общее число матриц

$$\chi^2(obs) = \frac{(F_M - 0.2888N)^2}{0.2888N} + \frac{(F_{M-1} - 0.5776N)^2}{0.5776N} + \frac{(N - F_M - F_{M-1} - 01336N)^2}{0.1336N}$$

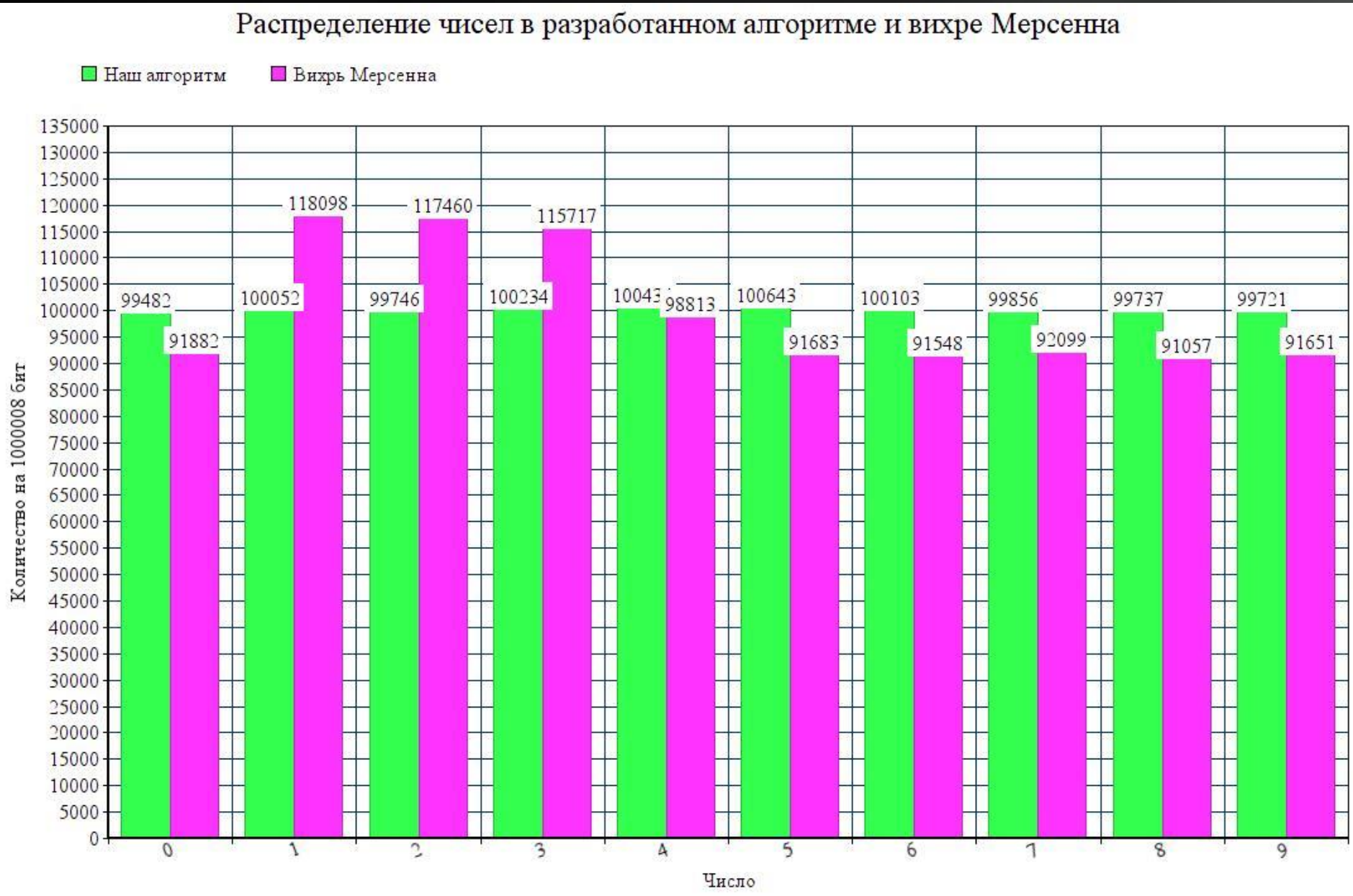
Результаты тестов NIST для разработанного алгоритма

RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

generator is <research_bin.dat>

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
0	2	1	1	1	1	0	1	2	1	0.911413	10/10	Frequency
0	3	2	1	2	0	0	0	1	1	0.350485	10/10	BlockFrequency
0	1	1	0	1	2	0	0	4	1	0.122325	10/10	CumulativeSums
0	3	0	1	2	1	0	0	1	2	0.350485	10/10	CumulativeSums
0	1	3	0	0	2	1	1	2	0	0.350485	10/10	Runs
0	3	0	1	1	1	0	1	2	1	0.534146	10/10	LongestRun
0	2	2	0	2	0	1	1	0	2	0.534146	10/10	Rank
0	1	2	0	0	2	3	0	0	2	0.213309	10/10	FFT
0	0	4	1	0	1	3	0	0	1	0.035174	10/10	NonOverlappingTemplate
0	2	1	1	1	1	0	0	1	3	0.534146	10/10	NonOverlappingTemplate
1	0	0	1	1	0	2	1	2	2	0.739918	10/10	NonOverlappingTemplate
0	1	1	2	1	1	1	1	1	1	0.991468	10/10	NonOverlappingTemplate
2	1	2	1	2	2	0	0	0	0	0.534146	9/10	NonOverlappingTemplate
0	0	3	0	1	1	0	0	3	2	0.122325	10/10	NonOverlappingTemplate
1	0	1	1	1	0	2	1	0	3	0.534146	10/10	NonOverlappingTemplate
2	2	0	2	0	2	0	1	1	0	0.534146	10/10	NonOverlappingTemplate
2	0	1	1	1	0	3	1	0	1	0.534146	10/10	NonOverlappingTemplate
0	2	1	2	1	1	0	1	1	1	0.911413	10/10	NonOverlappingTemplate
2	1	0	1	2	1	1	0	0	2	0.739918	10/10	NonOverlappingTemplate
0	0	2	0	2	1	2	0	1	2	0.534146	10/10	NonOverlappingTemplate

0	0	3	2	0	1	0	1	1	2	0.350485	10/10	NonOverlappingTemplate
1	1	2	0	2	1	0	1	0	2	0.739918	10/10	NonOverlappingTemplate
1	0	2	1	1	2	0	1	1	1	0.911413	10/10	NonOverlappingTemplate
3	0	2	1	1	2	1	0	0	0	0.350485	10/10	NonOverlappingTemplate
1	0	1	1	0	2	1	1	2	1	0.911413	10/10	NonOverlappingTemplate
1	0	0	3	1	1	2	0	2	0	0.350485	10/10	NonOverlappingTemplate
1	0	0	1	1	2	0	3	1	1	0.534146	10/10	NonOverlappingTemplate
0	0	3	2	1	2	1	0	0	1	0.350485	10/10	NonOverlappingTemplate
0	0	1	1	1	2	2	1	1	1	0.911413	10/10	NonOverlappingTemplate
1	0	0	3	2	2	0	0	0	2	0.213309	9/10	NonOverlappingTemplate
0	2	1	1	2	0	2	1	1	0	0.739918	10/10	NonOverlappingTemplate
0	1	2	1	3	0	0	0	2	1	0.350485	10/10	NonOverlappingTemplate
1	0	0	2	0	1	2	2	0	2	0.534146	10/10	NonOverlappingTemplate
0	1	0	0	2	2	0	2	2	1	0.534146	10/10	NonOverlappingTemplate
0	1	0	1	1	0	2	2	2	1	0.739918	10/10	NonOverlappingTemplate
1	0	0	0	1	1	1	0	2	4	0.122325	10/10	NonOverlappingTemplate
0	1	1	0	1	0	2	1	3	1	0.534146	10/10	NonOverlappingTemplate
2	1	0	1	0	2	1	1	2	0	0.739918	10/10	NonOverlappingTemplate
2	1	3	0	0	0	0	3	0	1	0.122325	10/10	OverlappingTemplate
0	0	0	0	0	0	0	10	0	0	0.000000 *	10/10	Universal
1	1	1	1	0	2	3	0	0	1	0.534146	10/10	ApproximateEntropy
1	1	0	1	1	1	2	0	2	1	0.911413	10/10	Serial
1	0	0	1	3	1	0	3	0	1	0.213309	10/10	Serial
1	1	1	1	3	2	1	0	0	0	0.534146	10/10	LinearComplexity

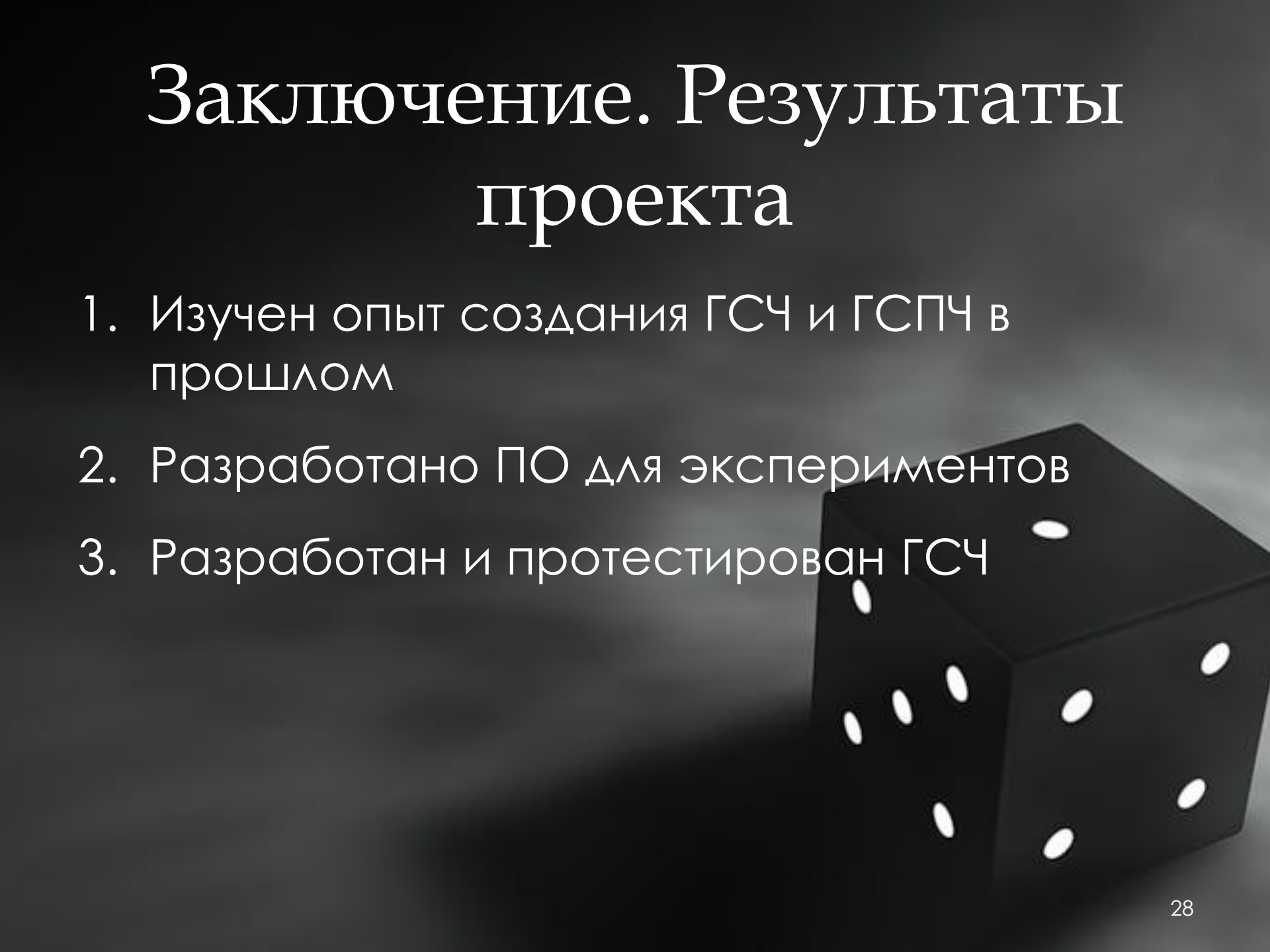

Распределение чисел в разработанном алгоритме и вихре Мерсенна
Наш алгоритм
Вихрь Мерсенна
Количество на 1000008 бит
Число
0
1
2
3
4
5
6
7
8
9
99482
91882
100052
118098
99746
117460
100234
115717
100643
98813
100643
91683
100103
91548
99856
92099
99737
91057
99721
91651

Сравнение разработанного алгоритма и вихря Мерсенна

	Частотный побитовый тест	Частотный блочный тест	Тест кумулятивных сумм	Тест рангов бинарных матриц	Спектральный тест	Тест на линейную сложность
Наш алгоритм	0.911413	0.350485	0.350485	0.534146	0.213309	0.534146
Вихрь Мерсенна	0.000000	0.000000	0.000000	0.350485	0.534146	0.008879

Заключение. Результаты проекта

1. Изучен опыт создания ГСЧ и ГСПЧ в прошлом
2. Разработано ПО для экспериментов
3. Разработан и протестирован ГСЧ



Источники

1. There's Math.random(), and then there's Math.random().17 December 2015. [Электронный ресурс]. URL: https://v8.dev/blog/math-random (дата обращения: 24.10.2019).

2. Эффективная генерация числа в заданном интервале. [Электронный ресурс]. URL: https://habr.com/en/post/455702/ (дата обращения: 24.10.2019).

3. Случайные числа не случайны [Электронный ресурс]. URL: https://medium.com/@frontman/%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D0%B5-%D1%87%D0%B8%D1%81%D0%BB%D0%B0-%D0%BD%D0%B5-%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B-252e08e60828 (дата обращения: 24.10.2019).

4. Псевдослучайно vs. По-настоящему Случайно [Электронный ресурс]. URL: https://habr.com/en/post/137864/ (дата обращения: 24.10.2019).

5. NIST SP 800-22 A statistical test suit for Random and Pseudorandom Number Generators for Cryptographic Applications URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf (дата обращения: 17.11.2019)

6. Гадалка Шеннона [Электронный ресурс]. URL: https://sites.google.com/site/ltwood/projects/heshby/shannon (дата обращения: 17.11.2019)