

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Э. БАУМАНА

ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ

И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»



Моделирование

ЛАБОРАТОРНАЯ РАБОТА №2

КРИТЕРИЙ ПРОВЕРКИ ПОСЛЕДОВАТЕЛЬНОСТИ ЧИСЕЛ НА СЛУЧАЙНОСТЬ

Студент: Петухов И.С.

Группа: ИУ7-71

Преподаватель: Рудаков И.В.

Москва, 2016

Содержание

1	Аналитический раздел	3
2	Конструкторский раздел	4
3	Технологический раздел	5
3.1	Язык программирования	5
3.2	Примеры кода	5
3.3	Взаимодействие с пользователем	6
	Список использованных источников	7

1 Аналитический раздел

Цель данной работы - реализовать критерий проверки случайности последовательности. Сравнить результаты работы данного критерия на табличных случайных числах и случайных числах, сгенерированных алгоритмически (отдельно для одноразрядных, двухразрядных и трехразрядных). Так же необходимо предусмотреть возможность задания случайной последовательности вручную.

За эталон генератора случайных чисел (ГСЧ) принят такой генератор, который порождает последовательность случайных чисел с равномерным законом распределения в интервале $(0; 1)$. [1]

Примером физических ГСЧ могут служить: монета («орел» — 1, «решка» — 0); игральные кости; поделенный на секторы с цифрами барабан со стрелкой; аппаратный генератор шума (ГШ), в качестве которого используют шумящее тепловое устройство, например, транзистор. [1]

В качестве критериев будут использоваться тесты NIST [2]

2 Конструкторский раздел

Алгоритмическая последовательность случайных чисел создается встроенным в язык генератором случайных чисел.

Примечание: метод `Math.random()` не предоставляет криптографически стойкие случайные числа. Не используйте его ни для чего, связанного с безопасностью. Вместо него используйте `Web Crypto API` (API криптографии в вебе) и более точный метод `window.crypto.getRandomValues()`.

Табличная последовательность случайных чисел создается с помощью `/dev/urandom` символьным устройством ОС Linux [3]

Критерии проверки ПСЧ на случайность:

- частотный побитовый тест
- тест на одинаковые идущие подряд биты.

В связи с тем, что тесты поразрядные, то числа нужно генерировать с учетом старших нулей. Т.е. если формируем трехразрядные числа, то сгенерированное число 2 (10 в двоичной системе), нужно подать на тест в виде "010".

В программе реализовано **два режима генерации чисел** алгоритмическим способом.

- десятичный - (числа [0, 9], [10, 99], [100, 999]). Данная последовательность обречена на провал в данных тестах, так как количество "0" и "1" в числах из данных диапазонов в двоичном виде будет смещенным.

- двоичный - при нем генерируются числа в диапазонах (числа в двоичном виде) [0, 1], [00, 11], [000, 111].

Частотный побитовый тест Принимаем каждую «1» за +1, а каждый «0» за -1 и считаем сумму по всей последовательности. Очевидно, что чем более случайна последовательность, тем ближе это соотношение к 1. Данный тест оценивает, насколько это соотношение близко к 1. Вычисляем статистику, затем вычисляем Р-значение через дополнительную функцию ошибок. Если результат > 0.01 , то значит наша последовательность прошла тест. Рекомендуется тестировать последовательности длиной не менее 100 бит. [2]

Тест на одинаковые идущие подряд биты В тесте ищутся все последовательности одинаковых битов, а затем анализируется, насколько количество и размеры этих последовательностей соответствуют количеству и размерам истинно случайной последовательности. Смысл в том, что если смена 0 на 1 (и обратно) происходит слишком редко, то такая последовательность «не тянет» на случайную. [2]

3 Технологический раздел

3.1 Язык программирования

В качестве языка программирования выбран язык высокого уровня JavaScript.

3.2 Примеры кода

Листинг 3.1 — Частотный побитовый тест

```
1 function frequencyBitwiseTest(str) {
2
3     let s = 0;
4     let n = str.length;
5
6     for (let i = 0; i < n; ++i) {
7         if (str[i] === "1")
8             ++s;
9         else
10            --s;
11     }
12
13     let sobs = Math.abs(s) / Math.sqrt(n);
14
15     let p = erfc(sobs / Math.sqrt(2));
16
17     return p > 0.01;
18 }
```

Листинг 3.2 — Тест на одинаковые идущие подряд биты

```
1 function identicalConsecutiveBitstest(str) {
2     let s = 0;
3     let n = str.length;
4
5     for (let i = 0; i < n; ++i)
6         if (str[i] === "1")
7             ++s;
8
9     let pi = s / n;
10
11     if (Math.abs(pi - 1/2) >= 2 / Math.sqrt(n)) return false;
12
13     let v = 1;
14
15     for (let i = 1; i < n; ++i)
16         if (str[i] !== str[i - 1])
```

```

17         ++v;
18
19         let p = erfc(Math.abs(v - 2 * n * pi * (1 - pi)) / (2 * pi * (1 - pi)
20         * Math.sqrt(2 * n)));
21
22         return p > 0.01;
23     }

```

3.3 Взаимодействие с пользователем

Взаимодействие с пользователем осуществляется через html страницы, открытые в браузере. В пользовательском интерфейсе используются динамические таблицы.

Алгоритмический способ

встроенный в язык random

	NaN	NaN	NaN
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
test #1			
test #2			

Запуск бинарный режим

Запуск десятичный режим

Табличный способ

аппаратный способ получения чисел

	NaN	NaN	NaN
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
test #1			
test #2			

Запуск

Ручной ввод

	nums
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
test #1	
test #2	

Запуск

Рисунок 3.1 — Пример работы программы

Список использованных источников

1. *stratum.ac.ru*. Генераторы случайных чисел. — <http://stratum.ac.ru/education/textbooks/modelir/lection22.html>. — [Online; accessed 23-October-2016].
2. *Безопасности, компания Код*. Статистическая проверка случайности двоичных последовательностей методами NIST. — <https://habrahabr.ru/company/securitycode/blog/237695/>. — [Online; accessed 23-October-2016].
3. *@dlinyj, Сергей*. Создаём аппаратный генератор случайных чисел. — <https://habrahabr.ru/post/274833/>. — [Online; accessed 23-October-2016].