Козловский Александр Вячеславович – e-mail: kozlovskiy@sfedu.ru; кафедра вычислительной техники; аспирант.

Onishchenko Stefan Vladimirovich – Adygea State University; e-mail: osv@adygnet.ru; Maykop, Russia; the department of automated information processing and management systems; assistant.

Melnik Eduard Vsevolodovich – Southern Federal University; e-mail: evmelnik@sfedu.ru; Taganrog, Russia; dr. of eng. sc.

Kozlovsky Alexander Vyacheslavovich – e-mail: kozlovskiy@sfedu.ru; the department of computer engineering; graduate student.

УДК 004.8 DOI 10.18522/2311-3103-2022-5-97-105

Н.В. Драгныш

РЕАЛИЗАЦИЯ И ИССЛЕДОВАНИЕ ГЕНЕРАТОРА СЛУЧАЙНЫХ ЧИСЕЛ НА ОСНОВЕ ПАРАЛЛЕЛЬНОГО ПЕРЕМЕШИВАНИЯ

Использование генераторов случайных чисел находит широкое распространение. Программные генераторы как правило выдают псевдослучайные числа. Для улучшения характеристик, необходим источник "хаоса". В данной статье рассматривается реализация генератора случайных чисел, использующего идею перемешивания полного набора чисел в параллельном потоке. На работу процессора в таком режиме оказывают влияние множество случайных факторов, что позволяет выдавать случайные числа. У предложенного генератора каждое последующее число не зависит от предыдущего. Генератор может выдать любое число заданного диапазона, несмотря на то сколько и какие числа выпадали ранее. В том числе может выпасть с малой вероятностью одно и тоже число несколько раз подряд. Отсутствует заранее определенная последовательность выдачи случайных чисел. Период, свойственный программным генераторам псевдослучайных чисел, также отсутствует. Но при этом снижается быстродействие выдачи случайных чисел, в сравнении с программными генераторами псевдослучайных чисел. Целью исследования было оценить свойства предложенного генератора, в первую очередь статистические характеристики генерируемых чисел. В статье рассмотрены вопросы идеи параллельного перемешивания. Вопросы реализации на языке программирования. Представлен программный интерфейс класса MixRandomBase и особенности реализации класса MixRandomByte, использующего рабочий массив из байтов. Для проверки качества работы генераторов случайных и псевдослучайных чисел применяют различные тесты. Приведены результаты проверки работы генератора такими тестами как распределение на плоскости, тест равномерности, тест "стопка книг". Результаты тестирования позволяют судить о хороших статистических характеристиках разработанного генератора. Оценено быстродействие генератора. В сравнении с линейными конгруэнтными генераторами псевдослучайных чисел время выдачи последовательности чисел больше в сотни раз.

Генератор случайных чисел; перемешивание случайных чисел; параллельное программирование; тесты генераторов случайных чисел.

N.V. Dragnysh

IMPLEMENTATION AND STUDY OF THE RANDOM NUMBER GENERATOR BASED ON PARALLEL MIXING

The use of random number generators is widespread. Software generators usually produce pseudo-random numbers. To improve characteristics, it needs a source of "chaos". This article discusses the implementation of a random number generator that uses the idea of mixing a complete set of numbers in a parallel thread. The operation of the processor in this mode is influenced

by many random factors, which allows issuing random numbers. For the proposed generator, each subsequent number does not depend on the previous one. The generator can give out any number of the given range, regardless of how many and what numbers fell out earlier. Including the same number can fall out with a low probability several times in a row. There is no predetermined sequence of issuing random numbers. The period inherent in software pseudo-random number generators is also missing. However, at the same time, the speed of issuing random numbers decreases, in comparison with software pseudo-random number generators. The purpose of research was to evaluate the properties of the proposed generator, primarily the statistical characteristics of the generated numbers. The article deals with the idea of parallel mixing. Implementation issues in a programming language. The programming interface of the MixRandomBase class and implementation features of the MixRandomByte class, which uses a working array of bytes, are presented. Various tests are used to check the quality of the work of random and pseudo-random number generators. The results of checking the operation of the generator by such tests as the distribution on the plane, the uniformity test, the "stack of books" test are given. The test results make it possible to judge the good statistical characteristics of the developed generator. The speed of the generator is estimated. In comparison with linear congruent pseudo-random number generators, the time for issuing a sequence of numbers is hundreds of times longer.

Random number generator; random number mixing; parallel programming; random number generator tests.

Введение. Генераторы случайных и псевдослучайных чисел (ГСЧ и ГПСЧ) находят все большее применение во многих прикладных задачах. Изначально использовались в имитационном моделировании [1] (метод Монте-Карло), криптографии. Затем все активнее в многих областях программировании, системах и методах искусственного интеллекта. Случайные числа используются в том числе в педагогической деятельности [2, 3]. Неплохой обзор ГСЧ приведен в [1, 4, 5], также в [6–8]. Рекомендации построения генераторов в [9–11].

В статье [12] мною был предложен генератор случайных чисел, использующий идею перемешивания полного набора чисел в параллельном потоке.

Такой генератор содержит рабочий массив, заполненный всеми числами некоторого диапазона. Каждое число встречается один раз. Постоянно, в параллельном потоке, независимо от вызывающей программы и других программ, происходит перемешивание рабочего массива. Вспомогательным ГСЧ выбираются две позиции массива и эти элементы меняются местами. Когда необходимо выдать случайное число, тот же вспомогательный ГСЧ выбирает позицию и выдает соответствующие число вызывающей основной программе. В результате параллельной работы потока выдачи чисел и потока перемешивания, с учетом, что вспомогательный генератор для них один, получаются непредсказуемые постоянные сдвиги в алгоритме работы вспомогательного генератора.

У предложенного конечного генератора каждое последующее число не зависит от предыдущего, так как при выдаче генератор каждый раз работает с новым массивом чисел, число перемешиваний непредсказуемо. Генератор может выдать любое число заданного диапазона, несмотря на то сколько и какие числа выпадали ранее. В том числе может выпасть с малой вероятностью одно и тоже число несколько раз подряд. Отсутствует заранее определенная последовательность выдачи случайных чисел, так как выбору нового случайного числа генератором предшествует перемешивание рабочего массива, и количество замен до следующего выбора случайного числа не определено. Период, свойственный программным генераторам псевдослучайных чисел, также отсутствует.

Особенности реализации. Различные варианты описанного генератора были реализованы на языке программирования C#. Пробовались рабочие массивы разного размера (разрядность элементов от 8 бит до 64). В качестве вспомогательного генератора использовался комбинированный линейный конгруэнтный генератор,

код которого приведен в [1], а также встроенный генератор языка программирования (класс Random). При этом отличные статистические результаты получались уже на разрядности в 8 бит. С ростом размера массива качество сгенерированных чисел оставалось по прежнему отличным, но существенно увеличивались затраты на системные ресурсы. Похожие результаты получались и при рассмотренных вариантах вспомогательных генераторов. Поэтому в конечном варианте реализации использовался 8 битный рабочий массив и встроенный генератор в качестве вспомогательного.

Программный интерфейс генератора (MixRandomBase) содержит функции:

- NextByte() - возвращает один рандомный байт от 0 до 255;
- NextBytes(int size) – возвращает массив рандомных байтов;
- NextUInt16() – возвращает целое от 0 до UInt16.MaxValue;
- NextUInt32() – возвращает целое от 0 до UInt32.MaxValue;
- NextUInt64() – возвращает целое от 0 до UInt64.MaxValue;
- NextDouble() – возвращает вещественное число от 0 до 1 на основе 32 разрядного целого;
- NextDecimal() – возвращает вещественное число от 0 до 1 на основе 64 разрядного целого;
- StartMix() – старт параллельного перемешивания;
- StopMix() – остановка(пауза) параллельного перемешивания;
- SaveTxtFile(string filename) – сохранение внутреннего массива в текстовый файл;
- LoadTxtFile(string filename) – чтение внутреннего массива из текстового файла.

Класс MixRandomByte реализует этот интерфейс используя в качестве рабочего массив байт со значениями о 0 до 255. Программа, использующая генератор, создает объект этого класса. При создании экземпляра создается рабочий массив, инициализируется стартовыми значениями. Изначально этот экземпляр имеет неперемешанный рабочий массив. Перемешанный можно прочитать из файла функцией LoadTxtFile.

Функция StartMix запускает параллельный поток, в котором постоянно выполняется перемешивание рабочего массива:

- выполняется блокировка/захват (lock) вспомогательного генератора (чтобы избежать коллизий с основным потоком);
- вспомогательным генератором генерируются два числа от 0 до 255 – индексы элементов, которые меняются местами;
- освобождается вспомогательный генератор, чтобы мог использоваться основным потоком;
- блокируется рабочий массив (в процессе замены элементов есть момент времени, когда в рабочем массиве два одинаковых элемента, а один отсутствует);
- два элемента рабочего массива с выбранными индексами меняются местами;
- снимается блокировка с рабочего массива.

После старта перемешивания основная программа может вызывать функции возвращающие случайные числа. Функция NextByte блокирует вспомогательный генератор, берет от него индекс от 0 до 255, снимает блокировку и возвращает элемент рабочего массива с этим индексом. Остальные функции, возвращающие сгенерированные числа, используют необходимое количество вызовов NextByte и преобразуют в необходимый формат.

Когда основной программе не нужны случайные числа можно остановить перемешивание функцией StopMix. Текущее состояние рабочего массива можно сохранить в файл (SaveTxtFile).

Типичное использование генератора имеет вид:

```
…
MixRandomByte mr1 = new MixRandomByte();
mr1.LoadTxtFile("save.txt");
mr1.StartMix();
…
current = mr1.NextDouble();// вызываем случайные числа
…
mr1.StopMix();
mr1.SaveTxtFile("save.txt");
…
```

Таким образом, предложенный генератор случайных чисел на основе параллельного перемешивания (MixRandom) был программно реализован в виде класса C#. Этот класс может быть использован в любой программе, для которой необходимы более качественные (в статистическом смысле) последовательности случайных чисел.

Исследование характеристик. Для проверки качества работы генератора случайных и псевдослучайных чисел применяют различные тесты [5, 13–15]. Часто методы оценки качества генераторов разделяют на две группы:

- графические тесты – свойства последовательностей сгенерированных чисел отображаются в виде графических зависимостей, по виду которых делают выводы (человек-исследователь) о свойствах исследуемой последовательности;
- оценочные (статистические) тесты – статистические свойства последовательностей определяются числовыми характеристиками (критериями). На основе критериев делаются заключения о близости свойств анализируемой и случайной последовательностей.

К результатам работы рассматриваемого генератора были применены множество различных тестов. Все эти тесты успешно пройдены. Далее в статье приводятся результаты некоторых из них.

Тест (графический) *распределение на плоскости* [5, 13]. Предназначен для определения зависимостей между элементами сгенерированной последовательности (выборки). На квадратном поле наносятся точки с координатами (x_i, x_{i+1}), где x_i – элементы исследуемой выборки, $i = 0,1, \dots, n-2$, n – объем выборки. Затем анализируется полученная картинка. Если на поле присутствуют закономерности (фигуры, узоры, пятна, линии) – выборка не является случайной, элементы зависимы. На рис. 1 представлены результаты работы генератора перемешивания MixRandom.

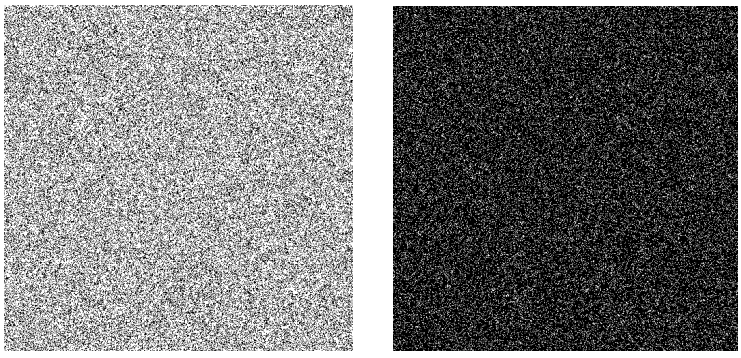

Рис. 1. Результаты теста распределение на плоскости

Поле справа заполнено точками выборки, объем которой больше в 5 раз, чем выборки поля слева.

Для сравнения на рис. 2 приведены результаты теста распределение на плоскости для неудачных реализаций ГСЧ [5, 13].

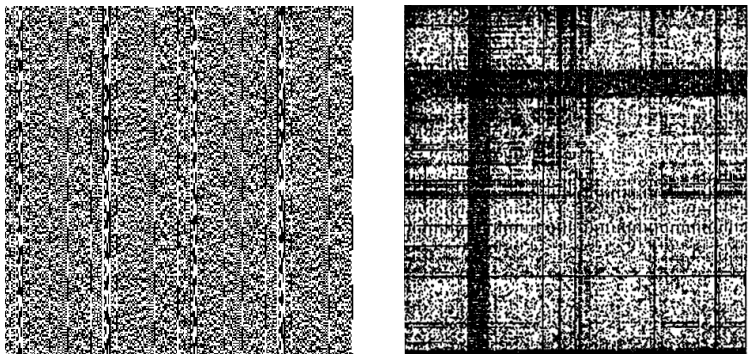

Рис. 2. Результаты теста распределение на плоскости для "плохих" ГСЧ

Для проверки на равномерность распределения применялся классический статистический *критерий* χ^2 (хи квадрат) Пирсона [16]. Сгенерированная выборка $x_1, x_2, \dots, x_n$ ($x_i \in [0; 1)$) разбивается на k интервалов одинаковой длины ($h = 1/k$). Количество интервалов определялось формулой Стерджесса: $k = 1 + \log_2 n$ (округляется до целого). Теоретическая вероятность попадания в интервал в случае равномерного закона, при одинаковой длине, одинаковая $p_j = 1/k$.

Наблюдавшееся значение критерия вычисляется по формуле

$$\chi^2_{\text{набл}} = \sum_{j=1}^{k} \frac{n_j^2}{n \cdot p_j} - n,$$

где n_j – частота попадания в интервал, n – объем выборки.

Наблюдавшееся значение критерия сравнивается с квантилями χ^2 распределения. Причем в случае тестирования ГСЧ применение критерия один раз ничего не дает (с определенной вероятностью гипотеза о равномерности примется или отклонится). Гораздо интереснее результаты многократного применения критерия к различным выборкам (проверок *N* выборок, каждая объемом *n*). И соответственно, сравнение результатов с квантилями χ^2 распределения.

В табл. 1 приведены результаты проверки генератора MixRandom на равномерный закон распределения с помощью критерия χ^2. Где *n* – объем одной сгенерированной выборки, *N* – число применений критерия (оно же количество сгенерированных выборок), *k* – число интервалов, $\chi^2_{\alpha;k-1}$ – α-квантиль критерия (α – уровень значимости, *k*-1 – число степеней свободы), Q_α – сколько раз из *N* применений критерия наблюдавшееся значение критерия $\chi^2_{\text{набл}}$ превысило $\chi^2_{\alpha;k-1}$. Проверялись квантили уровней $\alpha = 0{,}5$ и $\alpha = 0{,}05$. При *N*=1000 математическое ожидание превышения квантиля уровня 0,5 равно 500 ($0{,}5 \cdot 1000$), квантиля уровня 0,05 равно 50.

Таблица 1

Результаты проверки на равномерность

n	N	к	$\chi^2_{0{,}5;k-1}$	$\chi^2_{0{,}05;k-1}$	$Q_{0{,}5}$	$Q_{0{,}05}$
1000	1000	11	9,342	18,307	504	48
10000	1000	14	12,340	22,362	515	53
100000	1000	18	16,338	27,587	494	52

Результаты проверки хорошо согласуются с ожидаемыми результатами. $Q_{0,5}$ колеблется в районе 500, а $Q_{0,05}$ в районе 50.

Рябко Б.Я. и Пестуновым А.И. был предложен статистический тест *«Стопка книг»* [17]. В работах [18, 19] показана эффективность этого теста в сравнении с тестами NIST STS [14]. Тест позволил выявить проблемы многих генераторов, и в настоящее время считается одним из главных тестов генераторов случайных и псевдослучайных чисел [20, 21].

Пусть генератор порождает элементы из алфавита $A = \{a_1, a_2, \dots, a_S\}$, $S > 1$, и требуется по выборке $x_1, x_2, \dots, x_n$ проверить гипотезу H_0, заключающуюся в том, что элементы выборки независимы и вероятность появления любого символа алфавита одинаковая $(P(x_i = a_j) = 1/S,\ i = 1, \dots, n,\ j = 1, \dots, S,)$. Альтернативной гипотезы H_1 является отрицанием H_0.

Перед тестированием выборки в алфавите A фиксируется произвольный порядок, который меняется после анализа каждого выборочного элемента x_i следующим образом: буква x_i получает номер 1; номера тех букв, которые были меньше номера этой буквы, увеличиваются на 1; у остальных букв номера не меняются.

Формально эту процедуру можно описать так: пусть $\omega^i(a)$ это номер элемента $a \in A$ после анализа элементов $x_1, x_2, \dots, x_{i-1}$, тогда

$$\omega^{i+1}(a) = \begin{cases} 1, & x_i = a \\ \omega^i(a) + 1, & \omega^i(a) < \omega^i(x_i). \\ \omega^i(a), & \omega^i(a) > \omega^i(x_i) \end{cases}$$

Такая конструкция похожа на стопку книг, если считать, что номер книги совпадает с ее положением в стопке. Книга извлекается из стопки, после чтения кладется наверх, и ее номер становится первым. Книги, которые первоначально были над ней, двигаются вниз, а остальные остаются на месте.

Заметим, что $\omega^i(x_i)$ – номер элемента x_i в алфавите до обработки значения выборки x_i.

В отличие от многих других тестов, например, критерия хи квадрат, здесь подсчитывается не частота встречаемости букв в выборке, а частота встречаемости номеров букв при описанном упорядочивании.

Перед тестированием множество всех номеров $\{1, \dots, S\}$ разбивается на две непересекающиеся части: $A_1 = \{1, 2, \dots, \sqrt{S}\}$ и $A_2 = \{\sqrt{S} + 1, \dots, S\}$. Затем по выборке $x_1, x_2, \dots, x_n$ подсчитывается n_1 – количество номеров $\omega^i(x_i)$, принадлежащих подмножеству A_1, т. е. количество попаданий элементов в “верхнюю часть” “стопки книг”. Число $(n - n_1)$, очевидно, равно количеству попаданий в “нижнюю часть”. Далее вычисляется статистика [16] (которая при верности гипотезы H_0 должна быть распределена по закону χ^2 с одной степенью свободы)

$$\chi^2_{\text{набл}} = \frac{(n_1 - np_1)^2}{np_1} + \frac{\left((n - n_1) - n(1 - p_1)\right)^2}{n(1 - p_1)}.$$

После преобразований статистику можно привести к виду

$$\chi^2_{\text{набл}} = \frac{(n_1 - np_1)^2}{np_1(1 - p_1)}.$$

Так как, внутренняя реализация генератора MixRandom подразумевает естественный возврат одного случайного байта (остальные возвращаемые значения других типов собираются из байтов), алфавит при проверке был выбран $A = \{0, 1, \dots, 255\}$.

В табл. 2 приведены результаты проверки генератора MixRandom тестом «Стопка книг». Где n – объем одной сгенерированной выборки, N – число применений критерия (оно же количество сгенерированных выборок). Q_α – сколько раз из N применений критерия наблюдавшееся значение критерия $\chi^2_{\text{набл}}$ превысило $\chi^2_{\alpha;1}$. Квантиль χ^2 с одной степенью свободы уровня 0,5 $\chi^2_{0,5;1} = 0{,}455$, а уровня 0,05 $\chi^2_{0,05;1} = 3{,}841$. При N=1000 математическое ожидание превышения квантиля уровня 0,5 равно 500, квантиля уровня 0,05 равно 50.

Таблица 2

Результаты проверки тестом «Стопка книг»

n	N	$Q_{0,5}$	$Q_{0,05}$
1000	1000	516	41
10000	1000	502	45
100000	1000	499	52
1000000	1000	501	50

Результаты проверки хорошо согласуются с ожидаемыми результатами. $Q_{0,5}$ колеблется в районе 500, а $Q_{0,05}$ в районе 50.

Кроме статистических характеристик генератор MixRandom был проверен на быстродействие. Замерялось время генерации выборки заданного объема вещественных чисел типа double (8 байт). Результаты представлены в таблице 3. Приводится сравнение с встроенным ГСЧ класса Random.

Таблица 3

Результаты теста на быстродействие

	Время, мс			
n	1000	10000	100000	1000000
MixRandom	1	43	644	5467
Random	<1	<1	2	21

Как видно встроенный генератор выполняет выдачу в сотни раз быстрее. Что и неудивительно, так как он является линейным конгруэнтным генератором, выдающим результат на основе простой формулы от предыдущего числа. При этом числа встроенного генератора являются псевдослучайными, с неплохими статистическими характеристиками, но с периодом (пусть и очень большим) и с памятью (если числа появлялись чаще в некоторой части диапазона, то в будущем будут чаще появляться в тех где их было меньше).

Выводы. Таким образом, был реализован генератор случайных чисел MixRandom. Основанный на перемешивании массива байт в параллельном потоке от основной программы. Результаты тестирования позволяют судить о хороших статистических характеристиках последовательностей, выдаваемых генератором. Оценено быстродействие генератора. В сравнении с линейными конгруэнтными генераторами псевдослучайных чисел время выдачи последовательности чисел больше в сотни раз. Но реализация данного генератора была в первую очередь предназначена для оценки статистических характеристик. Для интенсивной работы генератора возможна модификация и оптимизация алгоритмов, которая улучшит быстродействие.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Кельтон В. Лоу А.* Имитационное моделирование. Классика CS. – СПб.: Питер; Киев: Издательская группа BHV, 2004. – 847 с.
2. *Драгныш Н.В.* Использование методов имитационного моделирования для преподавания курса «Теория вероятностей и математическая статистика» // Актуальные проблемы гуманитарных и естественных наук. – 2011. – № 12. – С. 26-29.
3. *Драгныш Н.В.* Использование инновационных технологий для преподавания курса "Теория вероятностей и математическая статистика" // Дискуссия. – 2010. – № 8. – С. 80-83.
4. *Кнут Д.Э.* Искусство программирования. Т. 2: Получисленные алгоритмы. – М.: Изд. дом “Вильямс”, 2019. – 832 с.
5. *Иванов М.А., Чугунков И.В.* Теория, применение и оценка качества генераторов псевдослучайных последовательеьностей. – М.: КУДИЦ-ОБРАЗ, 2003. – 240 с.
6. *L'Ecuyer P.* Tables of linear congruential generators of different sizes and good lattice structure // Math. of Comp. – 1999. – Vol. 68. – P. 249-260.
7. *Подорожный И.В.* Обзор аппаратных генераторов случайных чисел // Молодой ученый. – 2016. – № 1 (105). – С. 190-194.
8. *Задков В.Н., Владимирова Ю.В.* Классические и квантовые генераторы случайных чисел // Суперкомпьютеры. – 2013. – № 2. – С. 12-20.
9. Recommendation for Random Number Generation Using Deterministic Random Bit Generators. – URL: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf (дата обращения: 05.11.2022).
10. Recommendation for the Entropy Sources Used for Random Bit Generation. – URL: http://csrc.nist.gov/publications/drafts/800-90/sp800-90b_second_draft.pdf (дата обращения: 05.11.2022).
11. Recommendation for Random Bit Generator (RBG) Constructions. – URL: http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90c.pdf (дата обращения: 05.11.2022).
12. *Драгныш Н.В.* Построение генератора случайных чисел на основе параллельного перемешивания // Наука, техника и образование. – 2015. – № 6 (12). – С. 12-14.
13. *Григорьев А.Ю.* Методы тестирования генераторов случайных и псевдослучайных последовательностей // Ученые записки УлГУ. Сер. Математика и информационные технологии. – 2017. – № 1. – С. 22-28.
14. NIST Statistical Test Suite. – URL: http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html (дата обращения: 05.11.2022).
15. *Rukhin A. and others.* A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. NIST Special Publication 800-22 (with revisions dated May 15, 2001) NIST Statistical Test Suite. – URL: http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html (дата обращения: 05.11.2022).
16. Р 50.1.033–2001. Рекомендации по стандартизации. Прикладная статистика. Правила проверки согласия опытного распределения с теоретическим. Ч. I. Критерии типа хи-квадрат. – М.: Изд-во стандартов. 2002. – 87 с.
17. *Рябко Б.Я., Пестунов А.И.* “Стопка книг” как новый статистический тест для случайных чисел // Проблемы передачи информации. – 2004. – Т. 40. – Вып. 1. – С. 73-78.
18. *Пестунов А.И.* Теоретическое исследование свойств статистического теста “стопка книг” // Вычислительные технологии. – 2006. – Т. 11, № 6. – С. 96-103.
19. *Миненко А.И.* Экспериментальное исследование эффективности тестов для проверки генераторов случайных чисел // Вестник СибГУТИ. – 2010. – № 4. – С. 36-46.
20. *Ryabko B., Monarev V.* Using information theory approach for randomness testing // J. of Statistical Planning and Reference. – 2005. – Vol. 133, No. 1. – P. 95-110.
21. *Ryabko B., Stognienko V., Shokin Yu.* A new test for randomness and its application to some cryptographic problems // J. of Statistical Planning and Reference. – 2004. – Vol. 123, No. 2. – P. 365-376.

REFERENCES

1. *Kel'ton V. Lou A.* Imitatsionnoe modelirovanie. Klassika CS [Simulation modeling. CS classic]. Saint Petersburg: Piter; Kiev: Izdatel'skaya gruppa BHV, 2004, 847 p.
2. *Dragnysh N.V.* Ispol'zovanie metodov imitatsionnogo modelirovaniya dlya prepodavaniya kursa «Teoriya veroyatnostey i matematicheskaya statistika» [The use of simulation methods for teaching the course "Probability Theory and Mathematical Statistics"], *Aktual'nye problemy gumanitarnykh i estestvennykh nauk* [Actual problems of the humanities and natural sciences], 2011, No. 12, pp. 26-29.

3. *Dragnysh N.V.* Ispol'zovanie innovatsionnykh tekhnologiy dlya prepodavaniya kursa "Teoriya veroyatnostey i matematicheskaya statistika"[The use of innovative technologies for teaching the course "Probability Theory and Mathematical Statistics"], *Diskussiya* [Discussion], 2010, No. 8, pp. 80-83.
4. *Knut D.E.* Iskusstvo programmirovaniya. T. 2: Poluchislennye algoritmy [The art of programming. Vol. 2: Obtained algorithms]. Moscow: Izd. dom “Vil'yams”, 2019, 832 p.
5. *Ivanov M.A., Chugunkov I.V.* Teoriya, primenenie i otsenka kachestva generatorov psevdosluchaynykh posledovatele'nostey [Theory, application and evaluation of the quality of generators of pseudo-random sequences]. Moscow: KUDITS-OBRAZ, 2003, 240 p.
6. *L’Ecuyer P.* Tables of linear congruential generators of different sizes and good lattice structure, *Math. of Comp*., 1999, Vol. 68, pp. 249-260.
7. *Podorozhnyy I.V.* Obzor apparatnykh generatorov sluchaynykh chisel [Overview of hardware random number generators], *Molodoy uchenyy* [Young scientist], 2016, No. 1 (105), pp. 190-194.
8. *Zadkov V.N., Vladimirova Yu.V.* Klassicheskie i kvantovye generatory sluchaynykh chisel [Classical and quantum random number generators], *Superkomp'yutery* [Supercomputers], 2013, No. 2, pp. 12-20.
9. Recommendation for Random Number Generation Using Deterministic Random Bit Generators. Available at: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf (accessed 05 November 2022).
10. Recommendation for the Entropy Sources Used for Random Bit Generation. Available at: http://csrc.nist.gov/publications/drafts/800-90/sp800-90b_second_draft.pdf (accessed 05 November 2022).
11. Recommendation for Random Bit Generator (RBG) Constructions. Available at: http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90c.pdf (accessed 05 November 2022).
12. *Dragnysh N.V.* Postroenie generatora sluchaynykh chisel na osnove parallel'nogo peremeshivaniya [Building a random number generator based on parallel mixing], *Nauka, tekhnika i obrazovanie* [Science, technology and education], 2015, No. 6 (12), pp. 12-14.
13. *Grigor'ev A.Yu.* Metody testirovaniya generatorov sluchaynykh i psevdosluchaynykh posledovatel'nostey [Methods for testing generators of random and pseudo-random sequences], *Uchenye zapiski UlGU. Ser. Matematika i informatsionnye tekhnologii* [Uchenye zapiski UlGU. Ser. Mathematics and Information Technology. UlGU. Electron. Magazine], 2017, No. 1, pp. 22-28.
14. NIST Statistical Test Suite. Available at: http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html (accessed 05 November 2022).
15. *Rukhin A. and others.* A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. NIST Special Publication 800-22 (with revisions dated May 15, 2001) NIST Statistical Test Suite. Available at: http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html (accessed 05 November 2022).
16. R 50.1.033–2001. Rekomendatsii po standartizatsii. Prikladnaya statistika. Pravila proverki soglasiya opytnogo raspredeleniya s teoreticheskim. Ch. I. Kriterii tipa khi-kvadrat [R 50.1.033–2001. Recommendations for standardization. Applied Statistics. Rules for checking the agreement between the experimental distribution and the theoretical one. Part I. Chi-square tests]. Moscow: Izd-vo standartov. 2002, 87 p.
17. *Ryabko B.Ya., Pestunov A.I.* “Stopka knig” kak novyy statisticheskiy test dlya sluchaynykh chisel [“Pack of books” as a new statistical test for random numbers], *Problemy peredachi informatsii* [Problems. transfer of information], 2004, Vol. 40, Issue 1, pp. 73-78.
18. *Pestunov A.I.* Teoreticheskoe issledovanie svoystv statisticheskogo testa “stopka knig” [Theoretical study of the properties of the "stack of books" statistical test], *Vychislitel'nye tekhnologii* [Computational technologies], 2006, Vol. 11, No. 6, pp. 96-103.
19. *Minenko A.I.* Eksperimental'noe issledovanie effektivnosti testov dlya proverki generatorov sluchaynykh chisel [Experimental study of the effectiveness of tests for testing random number generators], *Vestnik SibGUT* [Vestnik SibSUTI], 2010, No. 4, pp. 36-46.
20. *Ryabko B., Monarev V.* Using information theory approach for randomness testing, *J. of Statistical Planning and Reference*, 2005, Vol. 133, No. 1, pp. 95-110.
21. *Ryabko B., Stognienko V., Shokin Yu.* A new test for randomness and its application to some cryptographic problems, *J. of Statistical Planning and Reference*, 2004, Vol. 123, No. 2, pp. 365-376.

Статью рекомендовал к опубликованию д.т.н. Н.Е. Сергеев.

Драгныш Николай Васильевич – Южный федеральный университет; e-mail: dragnysh@sfedu.ru; г. Таганрог, Россия; тел.: +79085110461; кафедра ИАСБ; к.т.н.; доцент.

Dragnysh Nikolay Vasilievich – Southern Federal University; e-mail: dragnysh@sfedu.ru; Taganrog, Russia; phone: +79085110461; the department IASB; cand. of eng. sc.; associate professor.