

## ОБЗОР И СРАВНИТЕЛЬНЫЙ АНАЛИЗ БИБЛИОТЕК ГЕНЕРАТОРОВ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ

В. В. Шахов

Институт вычислительной математики и математической геофизики СО РАН,  
630090, Новосибирск, Россия

---

УДК 004.421.5

Проводится обзор и сравнительный анализ двух широко известных библиотек генераторов псевдослучайных чисел от компаний Intel и Microsoft. В качестве основного критерия сравнения выбрана производительность генераторов.

**Ключевые слова:** генераторы псевдослучайных чисел, программное обеспечение.

At present simulation is general technique for performance analysis of complex systems. The main part of simulation is pseudorandom numbers generators. It allows to smooth differences between a model and real world. In this paper review of wide used software tools for pseudorandom numbers generating has been offered. Generators performance comparison has been made.

**Key words:** pseudorandom numbers generators, software development.

**Введение.** В настоящее время основным средством исследования сложных технологических и экономических систем является имитационное моделирование, с использованием которого решаются задачи оптимизации производственных процессов, оценивается отказоустойчивость роботизированных сборочных линий и конвейеров, исследуются режимы загрузки оборудования, отыскиваются эффективные стратегии комплексного управления логистическими процессами, проводится анализ финансово-экономического состояния предприятия, прогнозируются результаты различных управленческих решений и т. п. Важнейшими компонентами имитационных моделей, придающими им сходство с реальными явлениями, являются псевдослучайные числа (ПСЧ). Кроме того, ПСЧ находят широкое применение в численном анализе, тестировании алгоритмов, играх, сетевых протоколах и криптографии [1].

С ростом размерности приложений и увеличением ответственности разработчиков программных систем, использующих имитационное моделирование для проектирования и отладки механизмов управления крупными промышленными комплексами, ужесточаются требования к генераторам ПСЧ. Возникла потребность в разработке генераторов, которые кроме способности удовлетворять заданным статистическим свойствам должны также обладать высокой производительностью. Вероятно, указанная потребность сложилась прежде всего у программистов, активно работающих с языками C и C++. Действительно, язык C++ и его предшественник C широко применяются в системном программировании, на них пишутся операционные системы, драйверы, трансляторы, интерфейсы различного назначения, программы, которые взаимодействуют с операционной системой и т. д. Кроме того, к универсальности языка программирования добавляется богатый набор разнообразных библиотек, позволяющих быстро и эффективно решать различные задачи программирования. Отсюда можно сделать вывод, что языки C и C++ являются наиболее популярным инструментарием разработчиков промышленных программных продуктов, в частности автоматизированных

систем управления, описанных в работе [2]. Интерес к повышению эффективности генераторов ПСЧ у специалистов, пишущих на языках С и С++, подтверждается наличием соответствующего раздела в предварительной версии нового стандарта С++.

Таким образом, в данной работе прежде всего изучаются средства генерации ПСЧ, активно используемые программистами, пишущими на языках С и С++. Рассматривается библиотека векторной статистики (Vector Statistic Library (VSL)) от компании Intel, содержащая набор генераторов ПСЧ и являющаяся частью пакета MKL (Intel® Math Kernel Library). По сути, VSL является набором процедур, которые возвращают массив ПСЧ с заданными свойствами. Также рассматривается библиотека генераторов ПСЧ, расширяющая возможности инструментария С++ Visual Studio 2008 (полное название Standard C++ Library TR1 Extensions for Visual Studio 2008). В данной библиотеке (назовем ее Microsoft random) генераторы ПСЧ реализованы в виде шаблонов классов, интерфейсы которых соответствуют промежуточной версии стандарта С++ (документ ISO/IEC DTR 19768). Кроме того, исследуются стандартные средства получения ПСЧ, встроенные в большинство компиляторов С и С++, проводятся обзор основных возможностей библиотек Intel VSL и Microsoft random и сравнительный анализ библиотек.

### 1. Средства генерации ПСЧ. Рассмотрим стандартные средства получения ПСЧ.

1.1. *Стандартный генератор.* Современные средства разработки программного обеспечения, как правило, содержат стандартный генератор – генератор целочисленной равномерно распределенной псевдослучайной величины от нуля до некоторого большого значения. В качестве метода получения случайной последовательности традиционно использовался линейный конгруэнтный метод либо его частный случай – мультипликативный метод [1]. В настоящее время стремительно набирает популярность генератор ПСЧ, называемый вихрем Мерсенна (Mersenne twister), разработанный в 1997 г. японскими учеными М. Мацумото и Т. Нисимура [3]. Принцип работы генератора основывается на свойствах простых чисел Мерсенна, вихрь-преобразование, которое обеспечивает равномерное распределение ПСЧ в 623 измерениях. (Для сравнения, у линейных конгруэнтных генераторов оно не превышает 5, кроме того, они проигрывают в производительности.) Существует несколько реализаций алгоритма, различающихся размером используемого простого числа Мерсенна, наиболее распространенным из которых является MT19937, обеспечивающий большой период, равный числу Мерсенна  $2^{19937} - 1$ , что более чем достаточно практически для всех существующих приложений. Тем не менее появилась и критика генератора, в частности со стороны Д. Марсалья (George Marsaglia), являющегося признанным экспертом в области ПСЧ. Вихрь Мерсенна не является криптостойким генератором, что существенно затрудняет его использование в криптографии. Отмечалось, что генератор сложен в реализации. Метод генерации ПСЧ, предложенный Марсалья (multiply-with-carry), имеет простую реализацию, большой период и выигрывает в производительности. Однако появились и более эффективные реализации MT19937, исследования и усовершенствование генератора продолжаются. В то же время ряд исследователей в области методов Монте-Карло продолжает использовать хорошо изученный и статистически надежный линейный конгруэнтный метод, отмечая неудовлетворительные статистические свойства других генераторов. Таким образом, вопрос о наилучшем выборе генератора целочисленной равномерно распределенной псевдослучайной величины остается открытым и является делом вкуса.

Обращение к стандартному генератору может иметь вид

```
int rand ();
```

Получаем число от нуля до предопределенного большого числа (`RAND_MAX`). Однако стандартные генераторы, встроенные в компилятор, демонстрируют низкое качество [5]. В частности, младшие биты в случайном числе обладают плохими статистическими свойствами, поэтому `rand() % n` нельзя назвать хорошим способом генерирования случайных чисел от 0 до  $n - 1$ , более приемлемый результат получается, если предварительно получить ПСЧ с равномерным распределением от 0 до 1, т. е. использовать код следующего вида:

```
(double(rand()) / RAND_MAX) * n;
```

Генераторы ПСЧ с произвольным распределением, а также практически все генераторы случайных объектов (например, графов) основаны на использовании базового генератора, т. е. генератора ПСЧ, равномерно распределенных на интервале от 0 до 1. Источником для данных ПСЧ служит последовательность целочисленных равномерно распределенных псевдослучайных чисел, для генератора которой также будем использовать указанный термин. Последовательность ПСЧ, полученную с помощью базового генератора, будем называть базовой последовательностью.

Конструкция базовых генераторов предполагает определение начального значения псевдослучайной последовательности (инициализация генератора), от которого посредством рекуррентных вычислений строятся ПСЧ. Если начальное значение фиксировано при каждом обращении к генератору, то последовательность ПСЧ будет повторяться. Иногда такая ситуация устраивает пользователей, например при отладке программ. Однако часто каждое новое выполнение программы необходимо начинать с новым начальным значением. Инициализировать генератор ПСЧ некоторым числом ( $i$ ) можно следующим образом:

```
int srand (int i);
```

Начальное значение рекомендуется выбирать из окружения программы, часто для этого используются биты счетчика реального времени.

Таким образом, стандартные средства генерации ПСЧ ограничиваются возможностью инициализации генератора и доступом к базовой последовательности. Возможность, например, изменять параметры генератора или получать ПСЧ с распределением, отличным от равномерного, в стандартных средствах отсутствует.

1.2. *Библиотека VSL компании Intel Corporation.* Поставляемая компанией Intel библиотека VSL генераторов ПСЧ входит в пакет Intel Math Kernel Library, позиционируемый как средство для математических расчетов с высокой производительностью, реализуемых с использованием языков программирования FORTRAN, C и C++. Для использования функций библиотеки VSL в (C/C++)-программах необходимо в соответствующей среде разработки указать путь до заголовочного файла `mkl_vsl.h`, который содержится в папке `include` корневой директории пакета. По сути, библиотека является набором процедур, интерфейс которых выглядит следующим образом:

```
status = v<type>Rng<distribution>(method, stream, n, r, [<distribution parameters>]).
```

Здесь `v` – символ, индикатор того, что функция содержится в библиотеке VSL; `<type>` – тип генерируемых данных (используется один из трех символов: `s` : для типа `real`; `d` : для типа `double`; `i` : для типа `int`; префикс `Rng` указывает на то, что процедура является генератором ПСЧ с функцией распределения, название которой раскрывает параметр `<distribution>`; `method` – метод генерации ПСЧ с заданным распределением (метод обратной функции, метод отбраковки и т. п.); `stream` – идентификатор случайного потока

(их может быть несколько); **n** – размер выборки ПСЧ; **r** – имя массива, который хранит сгенерированную выборку; **status** – флаг ошибки VSL (выполнение процедуры прошло успешно, ошибок нет; параметры распределения заданы некорректно; ошибка в инициализации базового генератора; система не может выделить память; некорректный идентификатор случайного потока и т. д.); **<distribution parameters>** – список параметров функции распределения генерируемой случайной величины.

Следует отметить, что параметры обращения к генератору **n** и **r** являются независимыми, т. е. массив **r** должен быть описан заранее и содержать количество элементов, не меньшее, чем **n**.

Функции библиотеки можно разделить на три группы: методы выбора и инициализации базового генератора, генераторы для наиболее распространенных функций распределения, дополнительные возможности, заключающиеся в методах манипуляции потоками, средствах подключения пользовательских генераторов и обработки сгенерированных ПСЧ. Библиотека содержит следующие базовые генераторы:

- 32-битный мультипликативный генератор MCG(1132489760,  $2^{31} - 1$ ) [6];
- 32-битный генератор на основе виткового регистра сдвига с обобщенной отдачей GFSR(250,103) [7];
- комбинированный рекурсивный генератор MRG-32k3a [8];
- 59-битный мультипликативный генератор MCG( $13^{13}$ , 259) и генератор Уичмана-Хилла из библиотеки NAG Numerical Libraries [9];
- MT19937 [3];
- MT2203 [3];
- генератор квазислучайных чисел Соболев [10];
- генератор квазислучайных чисел Niederreiter [11].

Имеется также ряд генераторов для основных функций распределения, приведенных в таблице. Некоторые генераторы реализованы несколькими методами, пользователь может указать конкретный метод посредством выбора параметра “method”.

В приведенном ниже примере проводится оценка математического ожидания экспоненциально распределенной случайной величины по выборке из 10 000 значений.

```
#include ‘‘mkl_vsl.h’’
int main();
{
    double r[1000]; /* массив для хранения ПСЧ */
    double s; /* выборочное среднее */
    VSLStreamStatePtr stream;
    int i, j;
    /* Инициализация */
    s = 0.0;
    vslNewStream( &stream, VSL_BRNG_MT19937, 280775);
    /* Генерация */
    for ( i=0; i<10; i$++$ )
{
    vdRng Exponential (VSL_METHOD_DEXPONENTIAL_ICDF, stream, 1000, r, 1.0 );
    for ( j=0; j<1000; j$++$ )
```

## Наполнение библиотек генераторов ПСЧ

Блок	Возможности	Intel VSL	Microsoft random
Базовые функции	Выбор начального значения	Да	Да
	Линейный конгруэнтный метод	Да	Да
	Вихрь Мерсенна	Да	Да
	Метод Марсалья (multiply-with-carry)	Нет	Да
	Квазислучайные числа	Да	Нет
	Доступ к базовой последовательности	Нет	Да
Непрерывные распределения	Нормальное	Да	Да
	Равномерное	Да	Да
	Экспоненциальное	Да	Да
	Гамма	Да	Да
	Бета	Да	Нет
	Коши	Да	Нет
	Вейбулла	Да	Нет
	Логнормальное	Да	Нет
	Рэлея	Да	Нет
Лапласа	Да	Нет	
Дискретные распределения	Равномерное	Да	Да
	Пуассона	Да	Да
	Геометрическое	Да	Да
	Гипергеометрическое	Да	Нет
	Биномиальное	Да	Да
	Отрицательное биномиальное	Да	Нет

```

    {
        s += r[j];
    }
}
s /= 10000.0;
/* Удаление потока */
vslDeleteStream( &stream );
/* Вывод результата */
printf( "Sample mean of normal distribution = %f\n", s );
return 0;
}

```

Подробная информация о наполнении библиотеки и руководство по ее эксплуатации содержатся в работе [12].

1.3. *Библиотека random компании Microsoft Corporation.* Прототипом библиотеки Microsoft random можно считать соответствующую библиотеку в BOOST C++, которая изначально была основана на виртуальных функциях, а затем появился прототип с использованием шаблонов. После исправления ошибок, связанных с непортабельностью кода, обходом ограничений на инициализацию членов класса внутри объявления класса, улучшения интерфейса и компилятора в 2000 г. библиотека ПСЧ стала официальной частью BOOST C++. Предложения по улучшению интерфейсов библиотек были учтены экспертами комитета по стандартизации C++. В 2005 г. улучшенная версия интерфейсов вошла в предварительную версию (C++)-стандарта [13], известного также под названием TR1 (technical report 1).

Интерфейсы, предложенные в стандарте, нашли применение в указанной выше библиотеке генераторов ПСЧ, реализованной Microsoft. Отметим, что TR1 не содержит рекомендаций по выбору метода генерации случайных величин с заданным распределением, их выбор фиксирован и определяется предпочтениями программистов Microsoft. Библиотека находится в свободном доступе на сайте <http://msdn.microsoft.com>. Она снабжена инсталлятором, легко интегрируется с C++ Visual Studio 2008.

Набор базовых генераторов включает линейный конгруэнтный метод, вихрь Мерсенна, метод Марсалья. Пользователь может самостоятельно устанавливать параметры метода, а также использовать базовые генераторы с предустановленными рекомендуемыми параметрами, прошедшими проверку. Функции распределения, покрываемые генераторами библиотеки, перечислены в таблице. В следующем примере показан интерфейс шаблонного класса для генерации ПСЧ с распределением Пуассона:

```
template<class IntType = int, class RealType = double>
class poisson_distribution
{
public:
    typedef RealType input_type;
    typedef IntType result_type;
// конструктор и функции - члены класса
explicit poisson_distribution(const RealType& mean = RealType(1));
RealType mean() const;
void reset();
template<class UniformRandomNumberGenerator>
result_type operator()(UniformRandomNumberGenerator& urng);
};
```

Интерфейсы для генерации ПСЧ с другими законами распределений отличаются незначительно. Единственным классом, который не является шаблонным, является класс для распределения Бернулли.

Приведенный ниже код демонстрирует модель использования возможностей библиотеки на примере оценки математического ожидания экспоненциальной случайной величины.

```
#include <random>
#include <iostream>
int main()
{
    std::tr1::mt19937 U(280775);
    std::tr1::exponential_distribution<double> dist(1.0);
    double s = 0.0;
    for ( int i = 0; i < 10000; i$++$) {
        s+=dist(U);
    }
    s/=10000.0;
    std::cout << "sample mean = " << s << std::endl;
    return (0);
}
```

Здесь в качестве базового генератора используется вихрь Мерсенна MT19937, объем выборки равен 10 000 ПСЧ.

**2. Сравнительный анализ библиотек.** Выполним сравнительный анализ библиотек.

**2.1. Качественные показатели.** Поскольку библиотека Microsoft random разработана с использованием шаблонов классов, ее применение упрощает поддержку обобщенного программирования, т. е. программирования с использованием типов в качестве параметра при определении класса. Шаблон зависит только от свойств типа, передаваемого в качестве параметра, которые он явно использует, при этом не требуется, чтобы различные типы, используемые в качестве аргумента, были каким-либо образом связаны. Тем самым упрощается реализация безопасных, эффективных и легко настраиваемых компонент проектных решений, сокращается количество соответствующих строчек кода и уменьшается вероятность ошибок. Из приведенных выше примеров также следует, что библиотека дает возможность создания эффективных программ, не снижая при этом их читабельности или удобства наращивания.

Однако необходимость разработки приложений для разных платформ, обеспечения переноса программы с одного компилятора на другой делают предпочтительным выбор библиотеки VSL.

По доступности, включающей возможность получения информации о продукте и его приобретения через Интернет, сопровождение и консультации, ценовой сегмент, рассмотренные библиотеки находятся примерно на одном уровне.

**2.2. Функциональность.** Сравнительный анализ наполнения библиотек представлен в таблице. Следует уточнить, что VSL поддерживает лишь частную реализацию линейного конгруэнтного метода (мультипликативный метод). Под доступом к базовой последовательности понимается возможность изменять параметры базового генератора, например устанавливать любые значения для модуля, множителя и приращения для линейного конгруэнтного метода. Сюда же относится возможность манипулировать базовой последовательностью.

Заметим, что библиотека VSL поддерживает все распределения, указанные в предварительной версии стандарта C++.

**2.3. Производительность.** Как отмечалось выше, вопрос сравнения базовых генераторов является во многом субъективным. Практический интерес представляет сравнение генераторов ПСЧ для наиболее применимых распределений. Примем за метрику производительности количество ПСЧ, генерируемых за 1 с. В качестве базового генератора выбран MT19937. Рассматривались распределения, которые присутствуют в обеих библиотеках. Вычисления проводились на компьютере со следующими характеристиками: Intel Core 2 Duo CPU T7700 2.40 GHz, 3.00 GB RAM. При компиляции тестов для Microsoft random выбирался релиз-режим и устанавливались опции полной оптимизации (Configuration Properties/C/C++ / Optimization: Full Optimization). Параметры распределения выбирались таким образом, чтобы математическое ожидание было равно десяти. В случае распределения Бернулли значение параметра равно 0,7. Полученные результаты приведены на рис. 1. Видно, что производительность генераторов из VSL значительно выше, особенно в части дискретных распределений. Поскольку реализации методов получения ПСЧ с нормальным, экспоненциальным, равномерным распределениями и распределением Бернулли не могут существенно различаться, можно сделать вывод о том, что аппаратная оптимизация, задействованная в VSL, дает выигрыш от 10 до 50 %. Следовательно, проигрыш в 4-12 раз генераторов Microsoft random по производительности на дискретных распределениях объясняется не вполне удачным выбором метода генерации ПСЧ. Действительно, в результате анализа кода (файл random.h) для биномиального распределения выявлено, что генерация ПСЧ осуществля-

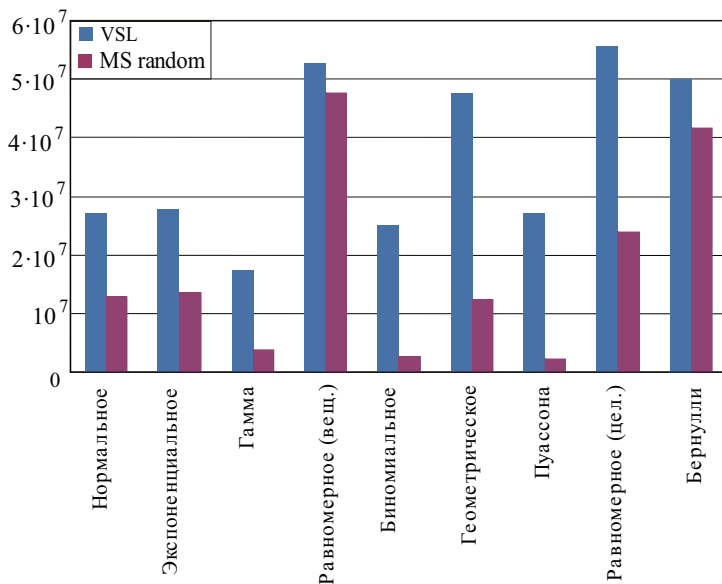


Рис. 1. Производительность генераторов ПСЧ (количество за 1 с)

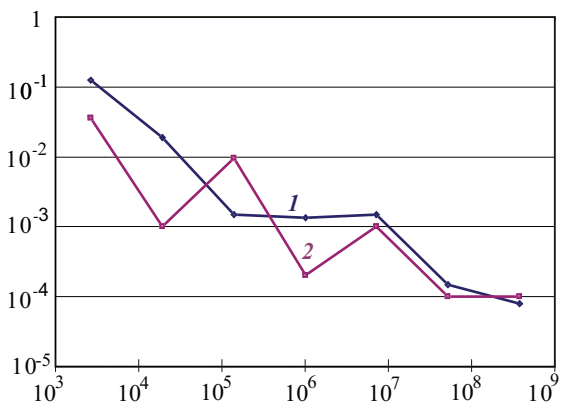


Рис. 2. Абсолютная погрешность оценки математического ожидания для биномиально распределенных ПСЧ: 1 – VSL, 2 – MS random

ется наиболее примитивным методом, основанным на определении биномиально распределенной случайной величины, т. е. для параметров распределения  $n$  и  $p$  проводится серия из  $n$  испытаний с вероятностью успеха  $p$  и подсчитывается количество успешных исходов. Для реализации соответствующего генератора в библиотеке VSL используется комбинированный метод на основе метода отбраковки и метода прямоугольника-клина-хвоста. Вообще, в VSL активно используется аппроксимация случайных величин, генерация которых трудоемка, случайными величинами, для которых существуют высокопроизводительные методы генерации. На малых выборках данный подход приводит к статистически менее надежным оценкам, но с увеличением выборки оправдывает себя.

На рис. 2 для биномиального распределения показан график модуля отклонения выборочного среднего от теоретически известного. Подобная тенденция наблюдается и для других распределений. Однако в VSL реализована возможность выбора метода генерации, который хотя и является менее быстрым, но на малых выборках позволяет получать более качественные оценки.

**Заключение.** Анализ производительности библиотек генераторов ПСЧ выявил существенное преимущество библиотеки VSL Intel по сравнению с Microsoft random. Кроме того, в целом наполнение VSL более представительно. Тем не менее на малых выборках или для вычислений, не требующих высокой производительности и точности, целесообразно задействовать библиотеку Microsoft random и использовать преимущества объектно ориентированного и обобщенного программирования.

## Список литературы

1. Кнут Д. Искусство программирования. Т. 2. Получисленные алгоритмы. 3-е изд. М.: Издат. дом "Вильямс", 2000.
2. Окольнішніков В. В. Использование имитационного моделирования при разработке автоматизированной системы управления технологическими процессами Северомуйского тоннеля // Вычисл. технологии. 2004. Т. 9, № 5. С. 82-101.
3. MATSUMOTO M., NISHIMURA T. Mersenne Twister: A 623-dimensionally equidistributed uniform pseudo-random number generator // ACM Trans. Model. Comput. Simulat. 1998. V. 8, N 1. P. 3-30.

4. MARSAGLIA G., ZAMAN A. A new class of random number generators // Ann. Appl. Probability. V. 1, N 3. P. 462-480.
5. СТРАНСТРУП Б. Язык программирования C++. 3-е изд. М.: БИНОМ, 1999.
6. L'ECUYER P. Tables of linear congruential generators of different sizes and good lattice structure // Math. Comput. 1999. N 68. P. 249-260.
7. KIRKPATRICK S., STOLL E. A very fast shift-register sequence random number generator // J. Comput. Phys. 1981. V. 40. P. 517-526.
8. L'ECUYER P. Good parameter sets for combined multiple recursive random number generators // Oper. Res. 1999. V. 47, N 1. P. 159-164.
9. NAG Numerical Libraries. [Electron. resource]. [http://www.nag.co.uk/numeric/numerical\\_libraries.asp](http://www.nag.co.uk/numeric/numerical_libraries.asp).
10. ЛЕВИТАН Ю. Л., СОБОЛЬ И. М. О датчике псевдослучайных чисел для персональных компьютеров // Мат. моделирование. 1990. № 2. С. 119-126.
11. BRATLEY P., FOX B. L., NIEDERREITER H. Implementation and tests of low-discrepancy sequences // ACM Trans. Model. Comput. Simulat. 1992. V. 2, N 3. P. 195-213.
12. Intel Math Kernel Library. Reference Manual. January, 2009.
13. International Standard ISO/IEC DTR 19768 // Draft Tech. Rep. on C++ Library Extensions.

*Владимир Владимирович Шахов – канд. физ.-мат. наук, науч. сотр.  
Института вычислительной математики и математической геофизики;  
e-mail: shakhov@rav.sccc.ru  
Дата поступления – 04.12.09*