УДК 004.42:519.2

Шевцов М. В., *здобувач вищої освіти 2 курсу спеціальності 122 Комп'ютерні науки,*
Комаров В. Ф., *канд. техн. наук, старший викладач кафедри інформаційних технологій*

ГЕНЕРАЦІЯ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ У СУЧАСНІЙ C++

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасному світі програмування генерація псевдовипадкових чисел відіграє ключову роль у різноманітних областях, від розробки ігор до криптографії та моделювання. Їх використання є необхідним елементом для створення випадкових подій та реалізації алгоритмів, які вимагають елемента випадковості.

Стандартна можливість генерування випадкових чисел з'явилася ще в мові програмування C, а потім була успадкована C++. Але можна зробити припущення, що зі стрибком технологій вимоги до стандартного алгоритму стали вищими. Це призвело до того, що функцію стали вважати застарілою і недостатньо ефективною. Але що саме не влаштувало сучасних кодерів?

Припустимо, що розробник хотів обрахувати суму мільйона випадкових елементів і був обмежений використанням функції std::rand(). В такому випадку розробник зіткнувся б із кількома проблемами. Запустивши цей код декілька разів, він міг помітити, що сума не дуже змінюється від запуску до запуску і завжди дорівнює значенню, близькому до 16'000, яке є достатньо малим для суми з 1 000 000 елементів. Проблема полягала у значенні макросу RAND_MAX, який визначає обмеження на максимальне число, яке дорівнює 32'767 (максимальному значенню змінної типу short integer). Це означає, що випадкове значення буде повторюватись кожні 32'767 ітерацій. До того ж сам алгоритм використовував лінійний конгруентний метод для обрахування наступного числа [1]. Такий метод є легко передбачуваним, тобто, принаймні, деякі біти згенерованого за його допомогою наступного числа доволі просто передбачити.

Усвідомивши перераховані проблеми, розробники почали шукати альтернативу, і так з'явився файл <random>, який став стандартною бібліотекою у стандарті C++11. Це значно потужніший інструмент, який містить 3 класи рушіїв випадкових чисел, кожен з яких реалізує в собі алгоритм, за допомогою якого обраховується наступний елемент. Для створення об'єкта рушія зазвичай необхідно вказувати багато шаблонних параметрів, що є максимально непрактичним. Тому були створені спеціальні псевдоніми зі вже підставленими найбільш використовуваними параметрами, які отримали назву генераторів.

На прикладі генератора std::mt19937 розглянемо переваги використання файлу <random>:

1. Максимальне значення, яке може бути згенеровано в такий спосіб, дорівнює 4'294'967'295 (тобто максимальному значенню 32-бітного числа) [2].

2. Обрахування наступного елементу відбувається значно швидше, ніж у std::rand().

3. Цей генератор використовує для генерації наступного числа алгоритм «Вихор Мерсенна», що ускладнює ймовірність передбачити це число [2].

4. Період повторення згенерованих чисел дорівнює $2^{19937} - 1$ [2].

Але одного лише генератора не достатньо: також потрібно обрати певний закон, що описуватиме значення випадкової величини та ймовірності їх появи [3]. Цей закон називається розподілом, і файл <random> містить реалізації найбільш популярних його видів, до яких входять 4 різновиди розподілу Бернуллі, 5 різновидів розподілів Пуассона, розподіли Фішера, Ст'юдента, Коші та багато інших. Є також 2 універсальні розподіли, які створюють цілі або дійсні значення, рівномірно розподілені в діапазоні. Щоб результати генерації не повторювались, як і у випадку функції std::rand(), треба задати «зерно», від якого буде відштовхуватись алгоритм під час генерації наступного числа. У мові С та версіях С++ до 11-го стандарту для цього часто використовували бібліотеку <ctime>, але після 11-го стандарту був доданий файл для сучасної роботи з часом під назвою <chrono>. В якості зерна можна використовувати звичайну константу типу unsigned, але підхід з файлом <chrono> підвищує непередбачуваність результатів.

Із прикладом застосування генераторів для рандомізованого алгоритму пошуку опорного елементу у сортуванні QuickSort, розробленого автором цієї роботи, можна ознайомитись у репозиторії [4].

Отже, ми розглянули методи генерування псевдовипадкових чисел у застарілому та сучасному стилях і визначили плюси та мінуси використання обох із них. Очевидно, що генерація з використанням std::rand() є значно передбачуванішою, має суттєві обмеження на період повторення та розмір генерованих чисел. З іншого боку, метод із використанням сучасних методів із файлів <random> та <chrono> надає істотно новий рівень контролю та непередбачуваності в процесі генерації випадкових чисел та робить період генерації повторень майже недосяжним.

Список використаних джерел

1. C++23 standard(ISO/IEC JTC1 SC22 WG21 N 4860).26.6 Random number generation.
2. Жильцов О. Б. Теорія ймовірностей та математична статистика у прикладах і задачах. Київський столичний університет імені Бориса Грінченка. 2015, 336 с.