Муниципальное бюджетное общеобразовательное учреждение
Энская средняя общеобразовательная школа

ИНДИВИДУАЛЬНЫЙ ПРОЕКТ

Реализация и сравнение генераторов псевдослучайных чисел

Выполнил: ученик 10 класса
`Aleph0x11`
Руководитель:
Учитель Информатики

2026 год

Содержание

ВВЕДЕНИЕ

Случайные числа играют крайне важную роль в информационных технологиях и используются в широком круге задач: от защиты данных до научных исследований. Получение истинно случайных чисел зачастую требует специального оборудования, поэтому в большинстве случаев используются псевдослучайные числа.

Псевдослучайное число — это число, которое выглядит как случайное, но таковым не являющееся, так как оно детерминировано и было получено в результате работы определённого алгоритма. Такое число не является истинно случайным и может быть воссоздано.

Генератор псевдослучайных чисел (ГПСЧ, англ. *pseudorandom number generator*, *PRNG*) — это алгоритм, создающий последовательность псевдослучайных чисел.

Зерно — это начальное значение, передаваемое в генератор псевдослучайных чисел.

Истинной случайности можно достичь, используя источник энтропии (например, тепловой шум) для получения случайного зерна.

ГПСЧ нашли широкое применение в различных задачах:

- в криптографии для генерации ключей шифрования;
- в лотереях, как британский ERNIE, используемый NS&I;
- в изучении случайных процессов методом Монте-Карло;
- для тестирования производительности и надёжности систем;
- в компьютерных играх для реализации процедурных локаций и случайных событий.

Цель работы: изучить принцип работы ГПСЧ, области их применения.
Задачи:

- рассмотреть самые используемые ГПСЧ;
- реализовать несколько ГПСЧ и сравнить их;

ГЛАВА I. ПРИМЕРЫ РАБОТЫ ГПСЧ

Одним из ранних ГПСЧ является **метод середины квадратов**, разработанный Джоном фон Нейманом. Суть этого метода заключается в следующем: чтобы получить случайное, например, десятизначное число, нужно возвести в квадрат десятизначное число, при необходимости дописав несколько нулей к началу, взять из середины квадрата десятизначное число, снова возвести его в квадрат и так далее. Например, сгенерируем четырёхзначное случайное число, начиная с 2815:

$$2815^2 = 07\overline{9242}25$$
$$9242^2 = 85\overline{4145}64$$
$$4145^2 = 17\overline{1810}25$$
$$1810$$

Недостатком этого метода является то, что последовательность может зацикливаться:

$$5000^2 = 25\overline{0000}00$$
$$0^2 = 00\overline{0000}00$$
$$0^2 = 0$$

Одним из современных алгоритмов является **линейный конгруэнтный метод** Д.Г. Лемера (англ. *linear congruential generator*, сокращённо *LCG*) задающийся этой формулой:

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

где a, m, $c \in \mathbb{Z}$ и $m > 1; m > a > 0; m > c > 0; m > X_0$. Последовательность этого метода тоже может зацикливаться, что является недостатком. Исходный код доступен в приложении А.

Другим подходом к генерации последовательности случайных чи-

сел является **регистр сдвига с линейной обратной связью** (англ. *linear feedback shift register*, сокращённо *LFSR*). LFSR работает по следующему принципу: существует **регистр**, состоящий из **ячеек памяти** (например, битов) с изначально заданным ненулевым значением; следующий бит генерируется определённой логической функцией, затем регистр сдвигается на один бит, и старшему биту присваивается значение сгенерированного бита. Например, следующая логическая функция [3] даёт максимальный **период** (количество неповторяющихся членов последовательности) в 65535 чисел 16-битному LFSR:

$$a = X_{16} \oplus X_{14} \oplus X_{13} \oplus X_{11},$$

где X_n — это n-ая ячейка памяти, а a — это новый старший бит регистра. Исходный код доступен в приложении А.

ГЛАВА II. ТЕСТИРОВАНИЕ

Протестируем и сравним ГПСЧ на основе линейного конгруэнтного метода и регистра сдвига с линейной обратной связью, реализованные мной (см. приложение А). Воспользовавшись утилитой ENT [2], получаем следующие результаты:

	χ^2	корреляция	энтропия на байт	метод Монте-Карло
LFSR	$0,07$	$-0,000048$	8	$3,131870773$
LCG	$123,15$	0.002521	$7,999831$	$3,140316545$

Примечание: χ^2 — тест хи-квадрат, идеал $\chi^2 \approx 255$, идеальная корреляция $= 0$, идеальная энтропия $= 8$, идеал метода Монте-Карло $= \pi \approx 3,14159$.

Рассмотрим результаты. LFSR показал хорошую корреляцию, энтропию и значение метода Монте-Карло, но очень низкую статистику χ^2. Это значит, что созданная им последовательность слишком равномерная и предопределённая. В то же время у LCG результаты тоже хороши, некоторые даже лучше чем у LFSR, например значение χ^2, которое гораздо ближе к идеалу и меньшее отклонение от π по методу Монте-Карло.

Для дальнейшего исследования LFSR и LCG была использована утилита STS (*the Statistical Test Suite for Random and Pseudorandom Number Generators*) [1] от Национального института стандартов и технологий США (NIST). LFSR прошёл 176 из 188 проверок и провалил 12. Из проваленных:

- Тест на ранг бинарной матрицы (англ. *Binary Matrix Rank Test*), т.к. это регистр с линейной обратной связью;
- Дискретное преобразование Фурье (англ. *Discrete Fourier Transform*), т.к. в файле размером 1МиБ, который сканировала утилита, последовательность повторилась 8 раз из-за низкого периода (65535 чисел);
- Тест на линейную сложность (англ. *Linear Complexity Test*), т.к. период LFSR оказался слишком коротким.

LCG прошел 183 теста из 188, не пройдя лишь 5. Не пройдены:

- Дискретное преобразование Фурье (англ. *Discrete Fourier Transform*);

- Тест на неперекрывающиеся шаблоны (англ. *Non-overlapping Template Matching*);
- Универсальный статистический тест Маурера (англ. *Maurer's Universal Statistical Test*);

Из вышеперечисленного можно сделать вывод, что случайность реализованных ГПСЧ приемлема для задач, которые **не требуют криптографической стойкости**.

ЗАКЛЮЧЕНИЕ

В ходе работы были достигнуты следующие результаты:

- **Рассмотрен принцип работы нескольких ГПСЧ:**
  - метод середины квадратов;
  - линейный конгруэнтный метод (LCG);
  - регистр сдвига с линейной обратной связью (LFSR).
- **Реализованы LCG и LFSR на языке C**. Исходный код представлен в приложении А. Генераторы используют типичные для своих типов параметры и ведут себя ожидаемым образом.
- **Проведено статистическое тестирование** с помощью утилиты ENT [2]. Результаты показали, что:
  - LFSR показывает такие хорошие качества, как высокая энтропия и нулевая корреляция, но низкое значение χ^2 свидетельствует об избыточной равномерности последовательности, вызванной малым периодом;
  - LCG показывает сопоставимые с LFSR результаты, а его статистика χ^2 даже превосходит χ^2 LFSR;
- **Проведено углублённое тестирование** набором NIST STS [1]:
  - LFSR прошёл 176 тестов из 188, непройденные проверки свидетельствуют о линейной структуре и малом периоде реализации;
  - LCG прошёл 183 тестов из 188;

Таким образом, поставленные в введении цели и задачи выполнены в полной мере.

Приложение А. Исходный Код

Представленные реализации ГПСЧ написаны на языке программирования C, и их работа протестирована в среде GNU/Linux.

A.1 Реализация LCG

Структура `lcg_t` хранит значения нужных для работы генератора коэффициентов, функция `init_lcg` задаёт им изначальные значения (в т.ч. зерно), функция `lcg` считает следующее число последовательности и возвращает его, записывая его в ранее созданную и инициализированную структуру.

Изначально эта реализация выдавала результат гораздо хуже из-за того, что максимально генератор генерирует 32-битные число, но возвращал и в тестовой программе шла запись в файл 64-битного числа — в итоге половина файла состояла из нулей.

```
#include <stdint.h>

typedef struct {
  uint32_t a;
  uint32_t c;
  uint64_t m;
  uint32_t x_n;
} lcg_t;

void init_lcg(int seed, lcg_t* gen){
  gen->a = 1664525;
  gen->c = 1013904223;
  gen->m = (1ULL << 32);
  gen->x_n = seed;
}

uint32_t lcg(lcg_t* gen){
    gen->x_n = (gen->a * gen->x_n + gen->c) % gen->m;
    return gen->x_n;
}
```

A.2 Реализация LFSR

Реализация 16-битного регистра с линейной обратной связью, объяснение принципа работы представлено в первой главе. Функция `lfsr` принимает указатель на регистр (16-битовое беззнаковое целое число), возвращает следующее число из последовательности и обновляет состояние регистра.

```
#include <stdint.h>

uint16_t lfsr(uint16_t* cylinder){
  uint16_t ret = 0x00;
  for(int i = 0; i < 16; i++){
    uint16_t bit = (*cylinder >> 15) & 1;
    ret |= (bit << i);
    uint16_t bit2 =
    ( *cylinder >> 0 ^
      *cylinder >> 2 ^
      *cylinder >> 3 ^
      *cylinder >> 5) & 1;
    *cylinder = *cylinder >> 1;
    *cylinder |= (bit2 << 15);
  }
  return ret;
}
```

A.3 Вспомогательная функция

Вспомогательная функция, которая использовалась при отладке LFSR. Принимает 16-битное беззнаковое целое число и возвращает его двоичное представление в виде строки.

```
#include <stdint.h>

const char* print_bits(uint16_t a){
  static char converted[17] = "";
  for(int i=0; i<16; i++){
    converted[15-i] = ((a >> i) & 1) ? '1' : '0';
  }
  converted[16] = '\0';
  return (const char*)converted;
}
```

A.4 Программа для тестирования

Программа, написанная, чтобы создать 2 файла размером 1МиБ каждый, состоящие из последовательностей двух реализованных ГПСЧ.

```
#include <stdint.h>
#include <stdio.h>
#include <sys/stat.h>
#include "include/libmyprng.h"

int main(void){
  const unsigned int LFSR_FILESIZE = (2 << 18);
  const unsigned int LCG_FILESIZE = (2 << 16);
  uint16_t cylinder = 0xaac8;
  uint16_t b = lfsr(&cylinder);
  struct stat dirstat;
  if(stat("samples", &dirstat)){
    mkdir("samples",0755);
  }
  FILE* fp = fopen("samples/lfsr_test.bin","w");
  if(!fp){
    perror("samples");
    return -1;
  }
  for(int i = 0; i < LFSR_FILESIZE; i++){
    fwrite(&b, sizeof(uint16_t), 1, fp);
    b = lfsr(&cylinder);
  }
  printf("Completed␣the␣LFSR␣test\n");
  fclose(fp);
  fp = fopen("samples/lcg_test.bin","w");
  lcg_t gen;
  uint64_t a;
  init_lcg(25565, &gen);
  for(int i = 0; i < LCG_FILESIZE; i++){
    a = lcg(&gen);
    fwrite(&a, sizeof(uint32_t), 1, fp);
  }
  printf("Completed␣the␣LCG␣test\n");
  fclose(fp);
  return (int)b;
}
```

Список литературы

[1] Rukhin A., Soto J., Nechvatal J. et al. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications [Электронный ресурс]. – URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf (дата обращения 18.03.2026).

[2] Walker J. A Pseudorandom Number Sequence Test Program [Электронный ресурс]. – URL: https://www.fourmilab.ch/random/ (дата обращения 17.03.2026).

[3] Koopman P. Maximal Length LFSR Feedback Terms [Электронный ресурс]. – URL: https://users.ece.cmu.edu/~koopman/lfsr/index.html (дата обращения 18.03.2026).