1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт компьютерных наук и кибербезопасности

**Высшая школа кибербезопасности**

**ЛАБОРАТОРНАЯ РАБОТА**

1. «**Выбор параметров ЛКГ и оценка случайности по критерию хи-квадрат и критерию перестановок**»
2. по дисциплине «Структуры данных»
3. Выполнил
4. Кондратьев Д. О., студент гр. 5131001/30003 \_\_\_\_\_\_\_\_\_\_\_\_

<*подпись*>

1. Преподаватель
2. Семьянов П. В., старший преподаватель \_\_\_\_\_\_\_\_\_\_\_\_

<*подпись*>

1. Санкт-Петербург
2. 2024

СОДЕРЖАНИЕ

[Цели и задачи 2](#_Toc160622236)

[Введение 2](#_Toc160622237)

[Теоретическая часть 3](#_Toc160622238)

[Ход работы 3](#_Toc160622239)

[Результаты 6](#_Toc160622240)

[Вывод 7](#_Toc160622241)

[Приложение 8](#_Toc160622242)

# Цели и задачи

Цель работы: научится выбирать правильные параметры и использовать случайные числа.

Задачи:

1. Выбрать правильные параметры линейного конгруэнтного генератора для данной длины машинного слова.
2. Определить на практике реальные характеристики (период и мощность) полученного генератора:
3. Оценить ЛКГ с помощью статистических критериев перестановок и хи-квадрат.

# Введение

Линейный конгруэнтный генератор (ЛКГ) является одним из наиболее распространенных методов генерации псевдослучайных чисел, который широко применяется в различных областях, начиная от компьютерной графики и моделирования до криптографии и статистики. Он основан на рекуррентном алгоритме, который генерирует последовательность чисел, выглядящую случайной, но фактически детерминированную.

Однако для обеспечения качественной работы ЛКГ необходимо тщательно выбирать его параметры. Выбор параметров в ЛКГ напрямую влияет на статистические свойства генерируемой последовательности чисел, такие как периодичность, равномерность распределения и другие. Неправильный выбор параметров может привести к появлению нежелательных структур или корреляций в последовательности, что делает ЛКГ непригодным для использования.

# Теоретическая часть

Формула ЛКГ в общем виде: xn + 1 = (a \* xn + c) % m

Где m – модуль ЛКГ, a – множитель, c – приращение.

Теорема 1.

Период ЛКГ равен m только тогда, когда:

1. c и m – взаимно простые числа
2. (a – 1) кратно любому простому p, которому кратно m
3. (a – 1) кратно четырём, если m кратно четырём

Характеристики ЛКГ:

1. Период
2. Разброс
3. Мощность

Мощность – такое минимальное число S, что (a – 1)S ≡ 0 (mod m). Существование S гарантируется, если выполняются условия теоремы 1.

# Ход работы

Для начала подберем параметры ЛКГ:

1. Выбор m:

Параметр m называется модулем линейного конгруэнтного генератора. От правильного выбора этого параметра зависит такая характеристика ЛКГ как период. Глядя на формулу ЛКГ, легко заметить, что количество всевозможных значений, которое может принимать очередное сгенерированное число x находится в промежутке от 0 до m – 1. Поэтому следует выбирать m достаточно большое, чтобы не сокращать потенциальный период генератора. Также рекомендуется брать m = 2k, где k – длина машинного слова компьютера, в этом случае остаток от деления на m будет находиться быстрее в силу особенностей устройства компьютера. В этой работе выберем параметр m для машинного слова длиной 32 бита, то есть m = 232 = 4294967296.

1. Выбор a:

Чтобы добиться максимального недостаточно лишь выбрать большое m. Необходимо выбрать сочетание параметров, удовлетворяющее теореме 1. Также рекомендуется брать такое *a*, чтобы *a mod 8 = 5* и *0.01m < a < 0.99m*. В данной работе выбран a = 53949677. Данное число удовлетворяет условиям теоремы 1.

1. Выбор c:

От выбора параметра c также зависят характеристики ЛКГ. В качестве приращения рекомендуется использовать нечетное число. Кроме того, в соответствии с теоремой для достижения периода равного m необходимо выбирать такой параметр c, чтобы числа c и m были взаимно простыми. В данной работе выбран параметр c = 47971, удовлетворяющий условиям теоремы 1.

Выбрав параметры, определим характеристики получившегося ЛКГ. Используем алгоритм Флойда, также известный как алгоритм зайца и черепахи. Период ЛКГ получается равный m, то есть 232, это означает, что условия теоремы 1 выполнены. Мощность определим, проходя в цикле от S = 1 и сравнивая (a – 1)S по модулю с m. Мощность получившегося генератора S = 17.

Далее проверим получившийся ЛКГ с помощью критерия хи-квадрат и критерия перестановок. Начальный член будем определять как функцию времени time(NULL).

* Критерий хи-квадрат

Разобьем диапазон возможных полученных значений на k = 21 интервал. Так как период ЛКГ m = 232, то длина каждого интервала будет d = 204522253 ( возьмём d = m / 21 + 1). Затем сгенерируем 10000 чисел и посчитаем количество чисел, попавших в каждый интервал, и воспользуемся формулой для оценки по критерию хи-квадрат:

A mathematical symbols and numbers

Description automatically generated with medium confidence

где ps – вероятность попадания в данный интервал (так как попадания в различные интервалы равновероятны, для всех интервалов p = 1/k); n – количество сгенерированных чисел; Ys – количество чисел, попавших в каждый интервал.

По получившемуся значению V , используя таблицу для значений хи-квадрат, судят о случайном/неслучайном поведении последовательности. Проведём 100 независимых наблюдений для различных сгенерированных последовательностей. Будем отбрасывать значения V меньшие 1%- или большие 99% точек согласно таблице хи-квадрат как недостаточно случайные. Если V лежит между 1%- и 5%-й точками или между 95%- и 99%-й точками, то эти числа "подозрительны"; если (интерполируя таблицу) V лежит между 5%- и 10%-й точками или 90%- и 95%-й точками, числа можно считать "почти подозрительными". Будем сравнивать значения V с табличными значениями хи-квадрат для количества степеней свободы ν = 20 (ν = k – 1).

* Критерий перестановок.

Сгенерируем 40000 чисел и разделим их на n = 10000 групп по t = 4 числа. Критерий перестановок заключается в оценке случайного поведения последовательности на основании частоты появления различных конфигураций распределений чисел в группах по величине. Определим конфигурацию каждой из групп по Алгоритму P:

A close up of a text

Description automatically generatedA close-up of a math problem

Description automatically generated with medium confidence

Затем посчитаем, сколько раз встретилась каждая из k=t!= 24 конфигураций. Вероятность появления каждой конфигурации равна p = 1/t! = 1/24. Применим формулу хи-квадрат для нахождения значения V. Проведем 100 независимых наблюдений для различных сгенерированных последовательностей. Будем отбрасывать значения V меньшие 1%- или большие 99% точек согласно таблице хи-квадрат как недостаточно случайные. Если V лежит между 1%- и 5%-й точками или между 95%- и 99%-й точками, то эти числа "подозрительны"; если (интерполируя таблицу) V лежит между 5%- и 10%-й точками или 90%- и 95%-й точками, числа можно считать "почти подозрительными". Будем сравнивать значения V с табличными значениями хи-квадрат для количества степеней свободы ν = 23 (ν = k – 1).

# Результаты

В результате работы были корректно выбраны параметры линейного конгруэнтного генератора, определены его характеристики и дана оценка ЛКГ с помощью статистического критерия хи-квадрат, а также критерия перестановок. A screenshot of a computer

Description automatically generatedПолученные результаты показывают, что генерируемые числа удовлетворяют используемым критериям.

# Вывод

В ходе работы я научился правильно подбирать параметры линейного конгруэнтного генератора, а также использовать случайные чисел.

# Приложение

#include <stdio.h>  
#include <math.h>  
#include <time.h>  
#include <stdlib.h>  
  
*// проверка на простое число*int is\_prime(unsigned long n) {  
 if (n == 2) return 1;  
 for (unsigned long div = 2; div <= sqrt(n) + 1; div++) {  
 if (n % div == 0) return 0;  
 }  
 return 1;  
}  
  
*// проверка на взаимную простоту*int is\_relatively\_prime(unsigned long a, unsigned long b) {  
 while ((a != 0) && (b != 0)) {  
 if (a > b) a %= b;  
 else b %= a;  
 }  
 if (a + b == 1) return 1;  
 else return 0;  
}  
  
*// ЛКГ*unsigned long lkg(unsigned long x) {  
 unsigned long m = 4294967296;  
 unsigned long a = 53949677;  
 unsigned long c = 47971;  
 return (a \* x + c) % m;  
}  
  
*// определение мощности ЛКГ*int find\_power(unsigned long a, unsigned long m){  
 unsigned long long n = 1;  
 unsigned long b = a - 1;  
 for (int s = 1; s < 10000; s++) {  
 n \*= b;  
 if (n % m == 0) {  
 return s;  
 }  
 }  
 return -1;  
}  
  
*// определение периода ЛКГ*unsigned long find\_period(unsigned long seed) {  
 unsigned long u1 = seed;  
 unsigned long u2 = lkg(seed);  
 while (u1 != u2) {  
 u1 = lkg(u1);  
 u2 = lkg(lkg(u2));  
 }  
 u1 = lkg(u1);  
 unsigned long period = 1;  
 *//printf("U2 = %u\n", u2);  
 //printf("U1 = %u\n", u1);* while (u1 != u2) {u1 = lkg(u1); period++;}  
 return period;  
}  
  
*// проверка ЛКГ по критерию хи-квадрат*double chi\_square\_criterion(unsigned long m, unsigned long seed) {  
 unsigned long x = seed;  
 int y[21], k = 21;  
 for (int i = 0; i < k; i++) y[i] = 0;  
 double d = m / k + 1;  
 int n = 10000;  
 for (int i = 0; i < n; i++){  
 x = lkg(x);  
 y[x / (int) d] += 1;  
 }  
 double p = 1.0 / k; *// вероятность попадания в каждый интервал одинакова и равна 1/t!  
 // применим критерий хи-квадрат* double V = 0;  
 for (int s = 0; s < k; s++) V += (y[s] \* y[s]) / p;  
 V = V / n - n;  
 return V;  
}  
  
*// проверка ЛКГ по критерию перестановок*double permutation\_criterion(unsigned long seed) {  
 unsigned long u[10000][4], x, buf;  
 x = seed;  
 int n = 10000, t = 4;  
 *// заполним n = 30 групп по t = 5 элементов* for (int i = 0; i < n; i++) {  
 for (int j = 0; j < t; j++) {  
 x = lkg(x);  
 u[i][j] = x;  
 }  
 }  
 *// вычислим для каждой группы значение f* int f[10000], r, max\_j;  
 for (int i = 0; i < n; i++) {  
 r = t;  
 f[i] = 0;  
 while (r > 1) {  
 *// ищем индекс максимума u[i][0:r]* max\_j = 1;  
 for (int j = 1; j < r; j++) {  
 if (u[i][j] > u[i][max\_j - 1]) {  
 max\_j = j + 1;  
 }  
 }  
 *// изменяем f* f[i] = r \* f[i] + max\_j - 1;  
 *// меняем местами u[i][r] и u[i][max\_j]* buf = u[i][max\_j - 1];  
 u[i][max\_j - 1] = u[i][r - 1];  
 u[i][r - 1] = buf;  
  
 r -= 1;  
 }  
 }  
 int y[24]; *// массив длиной t!, содержащий количество различных комбинаций* for (int i = 0; i < 24; i++) y[i] = 0;  
 for (int i = 0; i < n; i++) y[f[i]] += 1;  
 double p = 1.0 / 24; *// вероятность каждой комбинации одинакова и равна 1/t!  
 // применим критерий хи-квадрат* double V = 0;  
 for (int s = 0; s < 24; s++) V += (y[s] \* y[s]) / p;  
 V = V / n - n;  
 return V;  
}  
  
int main() {  
 *// Size of int is 32 bits  
 // Size of long is 64 bits  
 // Size of long-long is 64 bits* printf("----------------------- Выбор параметров -----------------------\n");  
 *// генерация начального члена* unsigned long x0;  
 srand(time(**NULL**));  
 x0 = time(**NULL**);  
 printf("Начальный член: %d\n", x0);  
  
  
 *// выбор параметров ЛКГ* unsigned long m = 4294967296;  
 unsigned long a = 53949677;  
 unsigned long c = 47971;  
 printf("модуль m: %llu, m простое: %d\n", m, is\_prime(m));  
 printf("приращение c: %d\n", c);  
 printf("множитель a: %llu\n", a);  
  
 printf("a mod 8 == %d\n", a % 8);  
  
 *// Теорема: 1) c и m - взаимно простые числа  
 // 2) (a - 1) кратно любому простому p, на которое делится m  
 // 3) (a - 1) кратно 4, если m кратно 4* if (is\_relatively\_prime(c, m)) printf("c и m - взаимно простые\n");  
 else printf("c и m - не взаимно простые\n");  
 if (is\_relatively\_prime(a, m)) printf("a и m - взаимно простые\n\n");  
 else printf("a и m - не взаимно простые\n\n");  
  
 *// определение мощности* printf("----------------------- Определение мощности -----------------------\n");  
 printf("Мощность ЛКГ = %d\n\n", find\_power(a, m));  
  
 *// определение периода* printf("----------------------- Определение периода -----------------------\n");  
 printf("Период ЛКГ = %llu\n\n", find\_period(x0));  
  
  
 *// проверка по хи-квадрат* int bad1 = 0; *// недостаточно случайные* int susp1 = 0; *// подозрительные* int almost\_susp1 = 0; *// почти подозрительные* int ok1 = 0; *// удовлетворительные* int bad2 = 0;  
 int susp2 = 0;  
 int almost\_susp2 = 0;  
 int ok2 = 0;  
 double V1, V2;  
 for (int i = 0; i < 100; i++) {  
 x0 = rand();  
 V1 = chi\_square\_criterion(m, x0);  
 V2 = permutation\_criterion(x0);  
  
 if (V1 <= 12.44 || V1 >= 28.41) {  
 if (V1 <= 10.85 || V1 >= 31.41) {  
 if (V1 <= 8.26 || V1 >= 37.57) bad1 += 1;  
 else susp1 += 1;  
 } else almost\_susp1 += 1;  
 }  
 else ok1 += 1;  
  
 if (V2 <= 14.85 || V2 >= 32.01) {  
 if (V2 <= 13.09 || V2 >= 35.17) {  
 if (V2 <= 10.19 || V2 >= 41.63) bad2 += 1;  
 else susp2 += 1;  
 } else almost\_susp2 += 1;  
 }  
 else ok2 += 1;  
  
 }  
 printf("----------------------- Критерий хи-квадрат -----------------------");  
 printf("\nПроверка разброса по табличным значениям хи-квадрат: (100 независимых наблюдений)\n"  
 "Отброшенные как недостаточно случайные значения (<1%% или >99%%): %d знач.\n"  
 "Подозрительные значения: %d знач.\n"  
 "Почти подозрительные значения: %d знач.\n"  
 "Удовлетворительные значения: %d знач.\n\n", bad1, susp1, almost\_susp1, ok1);  
 printf("----------------------- Критерий перестановок -----------------------");  
 printf("\nПроверка перестановок чисел по табличным значениям хи-квадрат: (100 независимых наблюдений)\n"  
 "Отброшенные как недостаточно случайные значения (<1%% или >99%%): %d знач.\n"  
 "Подозрительные значения: %d знач.\n"  
 "Почти подозрительные значения: %d знач.\n"  
 "Удовлетворительные значения: %d знач.\n", bad2, susp2, almost\_susp2, ok2);  
  
 return 0;  
}