**Лабораторная работа № 4**

**Реализация генератора псевдослучайной последовательности**

**1 Цель работы: изучить и закрепить умение генерирования алгоритмов работы генераторов псевдослучайных чисел.**

**2 Порядок выполнения работы:**

**2.1 Выбрать язык программирования.**

**2.2 Изучить материал.**

**2.3 Выполнить задания.**

**2.4 Ответьте на контрольные вопросы.**

**2.4.1 Где используются случайные числа?**

**2.4.2 ГСЧ – это…?**

**2.4.3 Отличия аппаратных и программных ГСЧ.**

**2.4.4 Случайное число – это….**

**2.5 Оформить отчет.**

**Теоретические сведенья:**

**Способы получения случайных чисел**

В программировании достаточно часто находят применение последовательности чисел, выбранных случайным образом из некоторого множества. В качестве примеров задач, в которых используются случайные числа, можно привести следующие:

* тестирование алгоритмов;
* имитационное моделирование;
* некоторые задачи численного анализа;
* имитация пользовательского ввода.

Для получения случайных чисел можно использовать различные способы. В общем случае все методы генерирования случайных чисел можно разделить на аппаратные и программные. Устройства или алгоритмы получения случайных чисел называют генераторами случайных чисел (ГСЧ) или датчиками случайных чисел.

Аппаратные ГСЧ представляют собой устройства, преобразующие в цифровую форму какой-либо параметр окружающей среды или физического процесса. Параметр и процесс выбираются таким образом, чтобы обеспечить хорошую «случайность» значений при считывании. Очень часто используются паразитные процессы в электронике (токи утечки, туннельный пробой диодов, цифровой шум видеокамеры, шумы на микрофонном входе звуковой карты и т.п.). Формируемая таким образом последовательность чисел, как правило, носит абсолютно случайный характер и не может быть воспроизведена заново по желанию пользователя.

К программным ГСЧ относятся различные алгоритмы генерирования последовательности чисел, которая по своим характеристикам напоминает случайную. Для формирования очередного числа последовательности используются различные алгебраические преобразования. Одним из первых программных ГСЧ является метод средин квадратов, предложенный в 1946 г. Дж. фон Нейманом.

Любые программные ГСЧ, не использующие внешних «источников энтропии» и формирующие очередное число только алгебраическими преобразованиями, не дают чисто случайных чисел. Последовательность на выходе такого ГСЧ выглядит как случайная, но на самом деле подчиняется некоторому закону и, как правило, рано или поздно зацикливается. Такие числа называются псевдослучайными.

Случайное число – число, представляющее собой реализацию случайной величины.

Детерминированный алгоритм – алгоритм, который возвращает те же выходные значения при тех же входных значениях.

Псевдослучайное число – число, полученное детерминированным алгоритмом, используемое в качестве случайного числа.

Физическое случайное число (истинно случайное) – случайное число, полученное на основе некоторого физического явления.

Генератор псевдослучайных чисел — [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC), порождающий последовательность [чисел](https://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%BE), элементы которой почти независимы друг от друга и подчиняются заданному [распределению](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9) (обычно [равномерному](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B5_%D1%80%D0%B0%D0%B2%D0%BD%D0%BE%D0%BC%D0%B5%D1%80%D0%BD%D0%BE%D0%B5_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5)).

**Линейный конгруэнтный генератор псевдослучайных чисел**

Генераторы псевдослучайных чисел могут работать по разным алгоритмам. Одним из простейших генераторов является так называемый линейный конгруэнтный генератор, который для вычисления очередного числа ki использует формулу

ki=(a\*ki-1+b)mod c,

где а, b, с — некоторые константы, a ki-1 — предыдущее псевдослучайное число. Для получения k1 задается начальное значение k0. Возьмем в качестве примера a=5,b=3,c=11 и пусть k0= 1. В этом случае мы сможем по приведенной выше формуле получать значения от 0 до 10 (так как с = 11). Вычислим несколько элементов последовательности:

k1 = (5 \* 1 + 3) mod 11 = 8;

k2 = (5 \* 8 + 3) mod 11 = 10;

k3 = (5 \* 10 + 3) mod 11 = 9;

k4 = (5 \* 9 + 3) mod 11 = 4;

k5 = (5 \* 4 + 3) mod 11 = 1.

Полученные значения (8, 10, 9, 4, 1) выглядят похожими на случайные числа. Однако следующее значение k6 будет снова равно 8:

k6 = (5 \* 1 + 3) mod 11 = 8,

а значения k7 и k8 будут равны 10 и 9 соответственно:

k7 = (5 \* 8 + 3) mod 11 = 10;

k8= (5 \* 10 + 3) mod 11 = 9.

Выходит, наш генератор псевдослучайных чисел повторяется, порождая периодически числа 8, 10, 9, 4, 1. К сожалению, это свойство характерно для всех линейных конгруэнтных генераторов. Изменяя значения основных параметров a, b и c, можно влиять на длину периода и на сами порождаемые значения ki. Так, например, увеличение числа с в общем случае ведет к увеличению периода. Если параметры a, b и c выбраны правильно, то генератор будет порождать случайные числа с максимальным периодом, равным c. При программной реализации значение с обычно устанавливается равным 2b-1 или 2b, где b — длина слова ЭВМ в битах.

Достоинством линейных конгруэнтных генераторов псевдослучайных чисел является их простота и высокая скорость получения псевдослучайных значений. Линейные конгруэнтные генераторы находят применение при решении задач моделирования и математической статистики, однако в криптографических целях их нельзя рекомендовать к использованию, так как специалисты по криптоанализу научились восстанавливать всю последовательность ПСЧ по нескольким значениям. Например, предположим, что противник может определить значения k0, k1, k2, k3. Тогда:

k1=(a\*k0+b) mod c

k2=(a\*k1+b) mod c

k3=(a\*k2+b) mod c

Решив систему из этих трех уравнений, можно найти a, b и c.

Для получения псевдослучайных чисел предлагалось использовать также квадратичные и кубические генераторы:

ki=(a12\*ki-1+a2\*ki-1+b) mod c

ki=(a13\*ki-1+a22\*ki-1+a3\*ki-1+b) mod c

Однако такие генераторы тоже оказались непригодными для целей криптографии по той же самой причине "предсказуемости".

**Метод Фибоначчи с запаздыванием**

Метод Фибоначчи с запаздываниями (Lagged Fibonacci Generator) — один из методов генерации псевдослучайных чисел. Он позволяет получить более высокое "качество" псевдослучайных чисел.

Наибольшую популярность фибоначчиевы датчики получили в связи с тем, что скорость выполнения арифметических операций с вещественными числами сравнялась со скоростью целочисленной арифметики, а фибоначчиевы датчики естественно реализуются в вещественной арифметике.

https://www.intuit.ru/EDI/01_08_18_2/1533075509-12654/tutorial/1011/objects/7/files/7_3.jpgИзвестны разные схемы использования метода Фибоначчи с запаздыванием. Один из широко распространённых фибоначчиевых датчиков основан на следующей рекуррентной формуле:

где ki — вещественные числа из диапазона [0,1], a, b — целые положительные числа, параметры генератора. Для работы фибоначчиеву датчику требуется знать max{a,b} предыдущих сгенерированных случайных чисел. При программной реализации для хранения сгенерированных случайных чисел необходим некоторый объем памяти, зависящих от параметров a и b.

Пример. Вычислим последовательность из первых десяти чисел, генерируемую методом Фибоначчи с запаздыванием начиная с k5 при следующих исходных данных: a = 4, b = 1, k0=0.1; k1=0.7; k2=0.3; k3=0.9; k4=0.5:

k5 = k1 - k4 = 0.7 - 0.5 = 0.2;

k6 = k2 - k5= 0.3 - 0.2 = 0.1;

k7 = k3 - k6 = 0.9 - 0.1 = 0.8;

k8 = k4 - k7 + 1 =0.5 - 0.8 + 1 = 0.7;

k9 = k5- k8 + 1 =0.2 - 0.7 + 1 = 0.5;

k10 = k6 - k9 + 1 =0.1 - 0.5 + 1 = 0.6;

k11 = k7 - k10 = 0.8 - 0.6 = 0.2;

k12 = k8 - k11 = 0.7 - 0.2 = 0.5;

k13 = k9 - k12 + 1 =0.5 - 0.5 + 1 = 1;

k14 = k10 - k13 + 1 =0.6 - 1 + 1 = 0.6.

Видим, что генерируемая последовательность чисел внешне похожа на случайную. И действительно, исследования подтверждают, что получаемые случайные числа обладают хорошими статистическими свойствами.

Для генераторов, построенных по методу Фибоначчи с запаздыванием, существуют рекомендуемые параметры a и b, так сказать, протестированные на качество. Например, исследователи предлагают следующие значения: (a,b) = (55, 24), (17, 5) или (97,33). Качество получаемых случайных чисел зависит от значения константы a: чем оно больше, тем выше размерность пространства, в котором сохраняется равномерность случайных векторов, образованных из полученных случайных чисел. В то же время с увеличением величины константы a увеличивается объём используемой алгоритмом памяти.

В результате значения (a,b) = (17,5) рекомендуются для простых приложений. Значения (a,b) = (55,24) позволяют получать числа, удовлетворительные для большинства криптографических алгоритмов, требовательных к качеству случайных чисел. Значения (a,b) = (97,33) позволяют получать очень качественные случайные числа и используются в алгоритмах, работающих со случайными векторами высокой размерности.

Генераторы ПСЧ, основанные на методе Фибоначчи с запаздыванием, использовались для целей криптографии. Кроме того, они применяются в математических и статистических расчетах, а также при моделировании случайных процессов. Генератор ПСЧ, построенный на основе метода Фибоначчи с запаздыванием, использовался в широко известной системе Matlab.

**Задания**

1. Выдайте на экран 10 случайных равномерно распределенных чисел в диапазоне:

* От 3 до 12, целые.
* Из множества {–3, 0, 6, 9, 12, 15}.
* От 3 до 12, вещественные.
* От –2,3 до 10,7 с шагом 0,1.
* Из множества {–30; 10; 63; 59; 120; 175}.
* Из множества {1; 0,1; 0,01; …; 10–15}.

2. Напишите программу, моделирующую игру «Быки и коровы». Программа выбирает с помощью датчика случайных чисел четырехзначное число с разными цифрами. Цель игры – угадать это число. На каждом шаге играющий называет четырехзначное число, а программа сообщает, сколько цифр числа угадано (быки) и сколько угаданных цифр стоит на нужном месте (коровы).