МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИНСТИТУТ МЕТАЛЛУРГИИ, МАШИНОСТРОЕНИЯ И ТРАНСПОРТА
КАФЕДРА «СОПРОТИВЛЕНИЕ МАТЕРИАЛОВ»

# Курсовая работа

Метод Фибоначчи с запаздыванием

Дисциплина: программирование на языке С

Разработал:
ст. гр. 33335/2 Комаров А.Е.

Преподаватель Ананьевский М.С.

Санкт-Петербург
2018

## Оглавление

## Введение

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

В отличие от генераторов, использующих линейный конгруэнтный алгоритм, фибоначчиевы генераторы можно использовать в статистических алгоритмах, требующих высокого разрешения.

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.

## История

Последовательность, в которой зависит более, чем от одного из предшествующих значений, и которая определяется следующей формулой: $X_{n+1} = (X_n + X_{n-1})\ mod\ (2^m)$, носит название последовательность Фибоначчи.

После, в начале 50-х годов, она была немного модернизирована: $X_{n+1} = (X_n + X_{n-k})\ mod\ (2^m)$, где $k$ рекомендовалось брать больше 15.

В 1958 году Митчел Дж. Ж. и Мур Д. Ф. вывели последовательность: $X_n = (X_{n-24} + X_{n-55})\ mod\ (2^m)$, где $n \geq 55$, $m$ — чётное число, $X_0, X_1, \dots, X_{54}$— произвольные целые числа. Числа 24 и 55 выбраны так, чтобы определялась последовательность, младшие значащие двоичные разряды $\left(X_n,\ mod\ (2)\right)$ которой имеют длину периода $2^{55} - 1$.

Очевидными преимуществами данного алгоритма являются его быстрота, поскольку он не требует умножения чисел, а также, длина периода, однако, случайность, полученных с помощью него чисел, мало исследована.

Числа 24 и 55 обычно называют *запаздыванием,* а числа $X_n$ — последовательностью Фибоначчи с запаздыванием.

Впоследствии, на основе этих работ было подобрано множество запаздываний, которые приводят к длинным периодам по модулю 2.

## Задача, которую решает метод Фибоначчи с запаздыванием

Нам нужно получить псевдослучайное значение. При чем, если нам нужна последовательность случайных чисел, то она должна обладать хорошими статистическими свойствами.

Известны разные схемы использования метода Фибоначчи с запаздыванием. Один из широко распространённых фибоначчиевых датчиков основан на следующей рекуррентной формуле:

$$k_i = \begin{cases} k_{i-a} - k_{i-b}, & \text{если } k_{i-a} \geq k_{i-b} \\ k_{i-a} - k_{i-b} + 1, & \text{если } k_{i-a} < k_{i-b} \end{cases},$$

где $k_i$ — вещественные числа из диапазона $[0\,,1)$, $a$ и $b$ — целые положительные числа, параметры генератора, называемые лагами. 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 $b$.

Такая формула будет выдавать нам случайные числа в диапазоне $[0\,,1)$, нам же нужны целые положительные числа в несколько разрядов. Модернизируем эту формулу в следующуу:

$$k_i = \begin{cases} k_{i-a} - k_{i-b}, & \text{если } k_{i-a} \geq k_{i-b} \\ k_{i-b} - k_{i-a}, & \text{если } k_{i-a} < k_{i-b} \end{cases},$$

где $k_i$ теперь целые положительные числа.

Для полученной формулы есть следующий алгоритм:

1. Запрашиваем у пользователя параметры $a$, $b$ и кол-во желаемых случайных величин ($Amount$).

2. Создаем массив $Arr[\,]$, размер которого будет равен $\max(a, b) + Amount + 1$.
3. Для работы этой формулы необходимо участок массива от 0 до $\max(a, b)$, заполнить случайными величинами. Как эти величины получены, не имеет значения. В данном случае будем использовать встроенную функцию $rand()$.
4. Выполняем цикл от $i = 0$, до $i = \max(a, b) + Amount$, увеличивая $i$ на единицу.
5. В каждой итерации цикла выполняем проверку:
   Если $Arr[i - a] \geq Arr[i - b]$,
   - следующий эл-т массива равен $Arr[i - a] - Arr[i - b]$,
   - в противном случае $Arr[i - b] - Arr[i - a]$.

**Разберем пример:**

Нужно получить 5 случайных целых положительных чисел. $a = 4$, $b = 7$.

Пусть $k_0 = 5$, $k_1 = 15$, $k_2 = 20$, $k_3 = 13$, $k_4 = 8$, $k_5 = 2$, $k_6 = 3$, $k_7 = 17$, тогда:

$k_8 = k_{8-7} - k_{8-4} = 15 - 8 = 7$ $(k_{i-b} > k_{i-a})$;

$k_9 = k_{9-7} - k_{9-4} = 20 - 2 = 18$ $(k_{i-b} > k_{i-a})$;

$k_{10} = k_{10-7} - k_{10-4} = 13 - 3 = 10$ $(k_{i-b} > k_{i-a})$;

$k_{11} = k_{11-4} - k_{11-7} = 17 - 8 = 9$ $(k_{i-a} \geq k_{i-b})$;

$k_{12} = k_{12-4} - k_{12-7} = 7 - 2 = 5$ $(k_{i-a} \geq k_{i-b})$;

$k_{13} = k_{13-4} - k_{13-7} = 18 - 3 = 15$ $(k_{i-a} \geq k_{i-b})$;

**В итоге** мы получили числа 7, 18, 10, 9, 5, 15. Видно, генерируемая последовательность чисел внешне похожа на случайную. Также видно, что диапазон выдаваемых случайных чисел лежит от 0, до максимального из выбранных в начале случайных чисел.

# Реализация на C++

```
#include <iostream>
#include <time.h>
#include <vector>
#include <iterator>
using namespace std;

typedef vector <int> Mass;

int main()
{
    unsigned int a = 0, b = 0;
    int Amount = 0;

    Mass::iterator LagA;                //создаем все итераторы
    Mass::iterator LagB;
    Mass::iterator Lag;

    std::cout<<" Lag a = ";             //запрос параметров
    std::cin>>a;
    std::cout<<" Lag b = ";
    std::cin>>b;
    std::cout<<" Amount of numbers ";
    std::cin>>Amount;

    Mass massive(1);                    //создаем массив
    massive.resize(max(a,b)+Amount+1);

    LagA=massive.begin();               //инициализируем лаги и сдвигаем на
    advance(LagA,a);                    //значения а и б
    LagB=massive.begin();
    advance(LagB,b);

    if(LagA>LagB) Lag=LagA;             //доп. итератор для корректной работы
    else Lag=LagB;                      //цикла

    Mass::iterator index;               //основной итератор движения по
    index=massive.begin();              //массиву

    srand(time(NULL));                  //заполняем первую часть массива
    while(index<=Lag){                  //случайными значениями
        *index = rand();
        cout<<"number="<<*index<<endl;
        index++;
    }

    for (; index != massive.end(); index++,LagA++,LagB++)
    {
        if (*LagA >= *LagB)             //получаем последовательность методом
        {                              // Фиблначчи с запаздываем
            *index = *LagA - *LagB;
        } else *index = *LagB -*LagA;
        std::cout<<*index<< std::endl;
    }
}
```

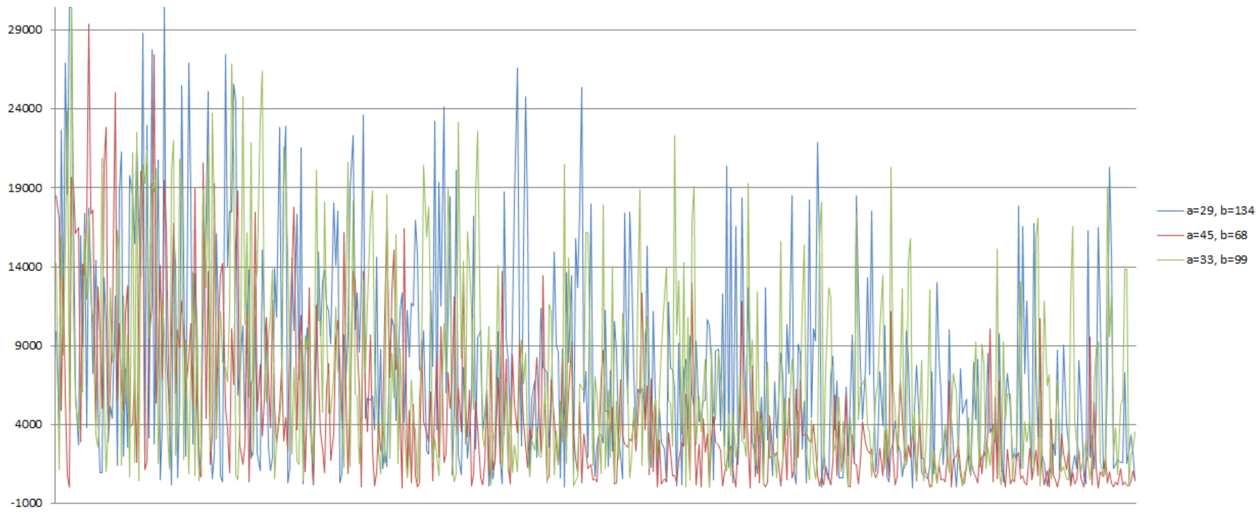
Пример выполнения полученной программы:

```
Lag a = 10
  10
 Lag b = 20
  20
 Amount of numbers 10
  10
3667
67
3944
5221
4249
29022
5554
3650
11114
12405
```

## Анализ алгоритма

В целом сложность алгоритма можно оценить как O(n), где n = $\max(a, b) + Amount + 1$.

Также, ниже приведен график полученных 500 значений при трех различных параметрах $a$ и $b$.




Из графика видно, что, действительно, данный метод обладает хорошими свойствами и выдает достаточно хорошие последовательности псевдослучайных чисел. Также, можно заметить, что при достаточно малой разности $b - a$ диапазон выдачи случайных чисел заметно сужается, что делает это метод неидеальным.

Лаги $a$ и $b$ — «магические» и их не следует выбирать произвольно. Рекомендуются следующие значения лагов: $(a, b) = (55,24), (17,5), (97,33)$.

Качество получаемых случайных чисел зависит от значения константы, а 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.

Значения $(a, b) = (17,5)$ можно рекомендовать для простых приложений, не использующих векторы высокой размерности со случайными компонентами. Значения $(a, b) = (55,24)$ позволяют получать числа, удовлетворительные для большинства 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. Фибоначчиев генератор случайных чисел (с лагами 20 и 5) используется в широко известной системе Matlab.

В настоящее время подобрано достаточно много пар чисел $a$ и $b$, приведём некоторые из них:

$$(38,89), (37,100), (30,127), (83,258), (107,378), (273,607), (1029,2281), \ldots$$

## Список литературы


- *Жельников В.* Криптография от папируса до компьютера. — 1996. — 325 с.
- https://www.intuit.ru/studies/courses/691/547/lecture/12383?page=2
- https://ru.wikipedia.org/wiki/Метод_Фибоначчи_с_запаздываниями
- http://nethash.ru/145-protivodejstvie-metodam-socialenoj-injenerii.html?page=9