Przegląd wybranych generatorów liczb pseudolosowych
i ich analiza pod kątem losowości.

MATEUSZ SOŁTYSIK, ANDRZEJ KWAK, WIKTOR DYNGOSZ
Politechnika Wrocławska, *Wydział Podstawowych Problemów Techniki*
May 28, 2014

Abstract

Keywords:

Wstęp

1 Opis i analiza poszczególnych algorytmów

1.1 Blum Blum Shub

Algorytm Blum Blum Shub został zaproponowany przez Lenore Blum, Manuela Blum oraz Michaela Shub'a w pracy pt. "A Simple Unpredictable Pseudo-Random Number Generator"[1].

```
private static final int p = 11;
private static final int q = 19;

public static long getRandomNumber() {
    seed = (seed * seed) % (p * q);
    return Math.abs(seed);
}
```

[1] http://epubs.siam.org/doi/abs/10.1137/0215025

1.2 Linear congruential generator

```
private final static long a = 25173;
private final static long b = 13849;
private final static long m = 32768;

public static long getRandomNumber() {
    seed = (a * seed + b) % m;
    return seed;
}
```

1.3 Mersenne twister

1.4 Park–Miller random number generator

```
private static final long max = ((long) 2 << 30) - 1;
private static final long a = 16807;

public static long getRandomNumber() {
    seed = (a * seed) % max;
    return seed;
}
```

1.5 Xorshift

Algorytm został stworony przez George Marsaglia[2]. Zasada działania opiera się na generowaniu następnych numerów wielokrotnie biorąc różnicę symetryczną z niego i przesuniętej bitowo wersji tej liczby.

```
public static long getRandomNumber() {
    seed ^= seed >> 12;
    seed ^= seed << 25;
    seed ^= seed >> 27;
    seed = (seed * 2685821657736338717L) % max;
    return Math.abs(seed);
}
```

Wnioski

[2]http://www.jstatsoft.org/v08/i14/paper