

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.
ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ
компьютерной безопасности и
криптографии

ТЕОРИЯ ПСЕВДОСЛУЧАЙНЫХ ГЕНЕРАТОРОВ

ОТЧЕТ ПО ПРАКТИЧЕСКОМУ КУРСУ

студента 4 курса 431 группы

факультета компьютерных наук и информационных технологий

Пензина Александра Сергеевича

Научный руководитель

доцент

И.И. Слеповичев

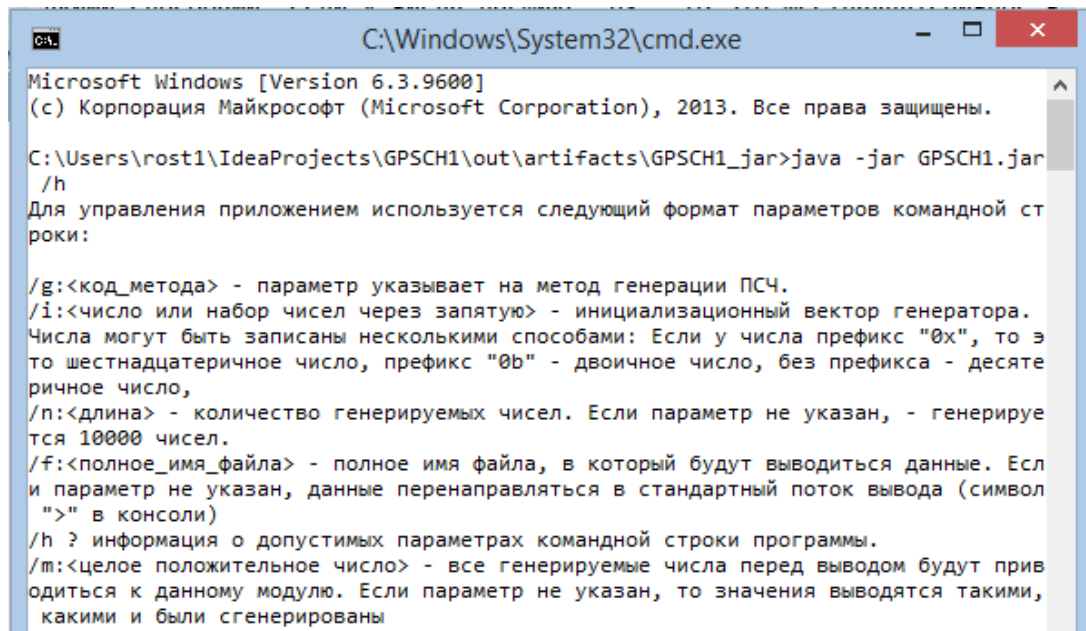
подпись, дата

Саратов 2020

1 ГПСЧ

Была реализована программа на языке Java для генерации последовательности псевдослучайных чисел различными методами и вывода её в консоль (файл). Основная логика запуска и выдачи результатов реализована в файле Main.java из прилагаемого архива.

Каждый генератор реализован в виде отдельного Java-класса – наследника абстрактного класса Generator, описанного в файле Main.java.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) Корпорация Майкрософт (Microsoft Corporation), 2013. Все права защищены.

C:\Users\rost1\IdeaProjects\GPSCCH1\out\artifacts\GPSCCH1_jar>java -jar GPSCCH1.jar
/h
Для управления приложением используется следующий формат параметров командной строки:

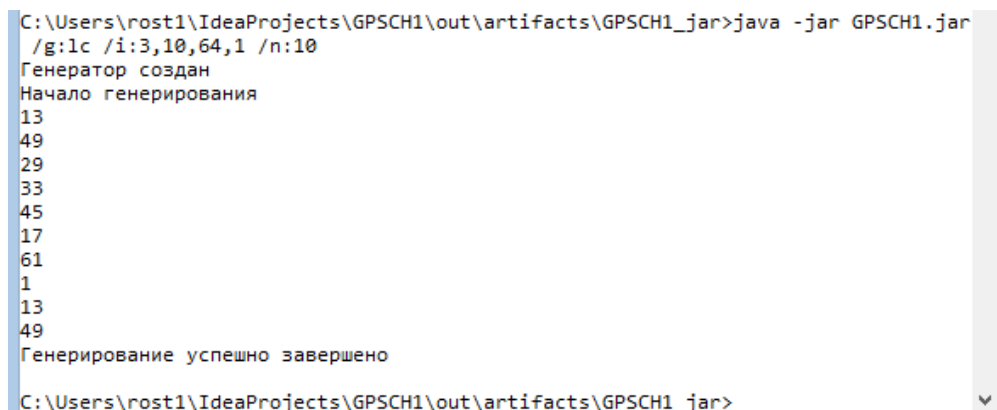
/g:<код_метода> - параметр указывает на метод генерации ПСЧ.
/i:<число или набор чисел через запятую> - инициализационный вектор генератора.
Числа могут быть записаны несколькими способами: Если у числа префикс "0x", то это шестнадцатеричное число, префикс "0b" - двоичное число, без префикса - десятичное число.
/n:<длина> - количество генерируемых чисел. Если параметр не указан, - генерируется 10000 чисел.
/f:<полное_имя_файла> - полное имя файла, в который будут выводиться данные. Если параметр не указан, данные перенаправляются в стандартный поток вывода (символ ">" в консоли)
/h ? информация о допустимых параметрах командной строки программы.
/m:<целое положительное число> - все генерируемые числа перед выводом будут приводиться к данному модулю. Если параметр не указан, то значения выводятся такими, какими и были сгенерированы
```

Рисунок 1 – Справка о параметрах командной строки.

1.1 Линейный конгруэнтный метод

Параметрами генератора являются множитель a , константа c и модуль m . Состояние — предыдущее число. Следующее число генерируется по формуле $X_{i+1} = (aX_i + c) \bmod m$.

Реализован в файле LC.java.



```
C:\Users\rost1\IdeaProjects\GPSCCH1\out\artifacts\GPSCCH1_jar>java -jar GPSCCH1.jar
/g:lc /i:3,10,64,1 /n:10
Генератор создан
Начало генерирования
13
49
29
33
45
17
61
1
13
49
Генерирование успешно завершено
C:\Users\rost1\IdeaProjects\GPSCCH1\out\artifacts\GPSCCH1_jar>
```

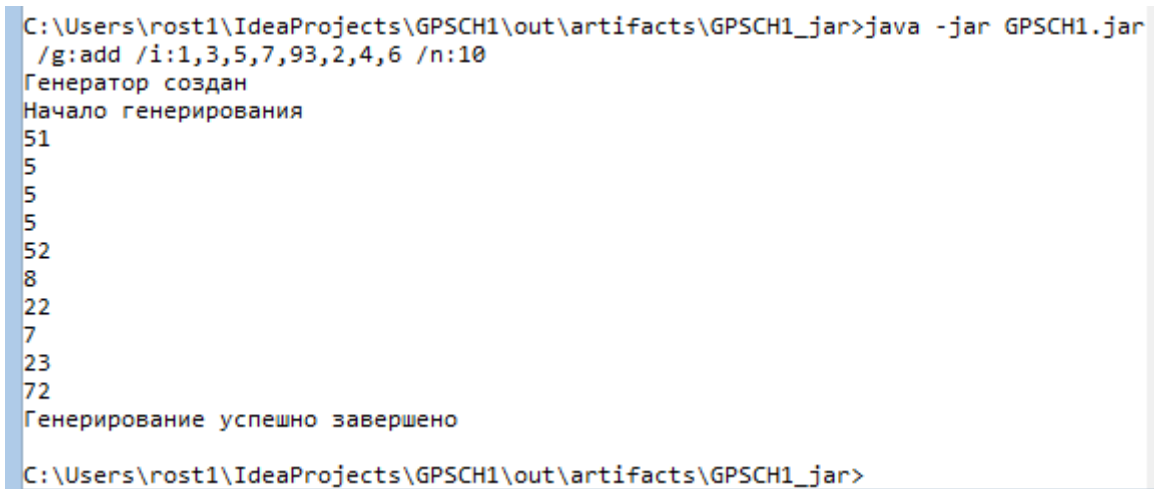
Рисунок 2 — Пример запуска линейного конгруэнтного генератора.

1.2 Аддитивный метод

Обобщение линейного конгруэнтного метода. Состоянием является s последних чисел. Параметры — s множителей a_i , константа c , модуль m . Следующее число генерируется по формуле:

$$x_{i+s+1} = \sum_{j=1}^s a_j * x_{i+j} + c \pmod{m}$$

Реализован в файле ADD.java.



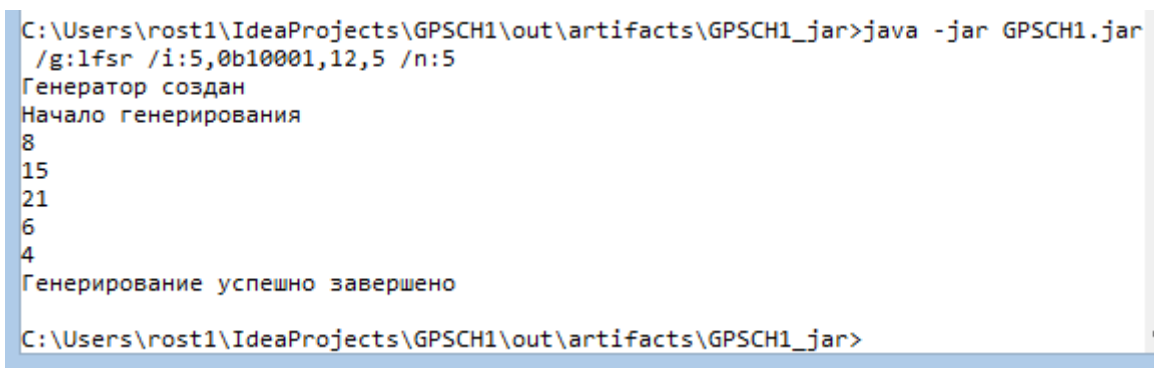
```
C:\Users\rost1\IdeaProjects\GPSC1\out\artifacts\GPSC1_jar>java -jar GPSC1.jar
/g:add /i:1,3,5,7,93,2,4,6 /n:10
Генератор создан
Начало генерирования
51
5
5
5
52
8
22
7
23
72
Генерирование успешно завершено
C:\Users\rost1\IdeaProjects\GPSC1\out\artifacts\GPSC1_jar>
```

Рисунок 3 - Пример запуска аддитивного генератора.

1.3 Регистр сдвига с линейной обратной связью

Генерируется последовательность битов. Состоянием является регистр из r последних битов. Следующий бит формируется как "исключающее или" битов регистра, находящихся на позициях j_1, \dots, j_m . После генерации нового бита содержимое регистра сдвигается в сторону старших разрядов, вместо младшего разряда поставляется полученный бит.

Реализован в файле LFSR.java.



```
C:\Users\rost1\IdeaProjects\GPSC1\out\artifacts\GPSC1_jar>java -jar GPSC1.jar
/g:lfsr /i:5,0b10001,12,5 /n:5
Генератор создан
Начало генерирования
8
15
21
6
4
Генерирование успешно завершено
C:\Users\rost1\IdeaProjects\GPSC1\out\artifacts\GPSC1_jar>
```

Рисунок 4 – пример запуска генератора на основе РСЛОС

1.4 Пятипараметрический метод

Частный случай РСЛОС. Генератор использует три параметра позиций q_1 , q_2 и q_3 . Следующий бит генерируется по формуле

$$x_{i+p} = x_i \oplus x_{i+q_1} \oplus x_{i+q_2} \oplus x_{i+q_3}.$$

Реализован в файле FIVER.java.

```
C:\Users\rost1\IdeaProjects\GPSCH1\out\artifacts\GPSCH1_jar>java -jar GPSCH1.jar
/g:5p /i:8,1,3,7,5,12 /n:5
Генератор создан
Начало генерирования
11
15
14
15
25
Генерирование успешно завершено

C:\Users\rost1\IdeaProjects\GPSCH1\out\artifacts\GPSCH1_jar>
```

Рисунок 5 – пример запуска пятипараметрического генератора

1.5 Нелинейная комбинация РСЛОС

Генератор состоит из k РСЛОСов. Для генерации следующего бита каждый РСЛОС генерирует бит (i ый РСЛОС генерирует x_i), после чего выходной бит определяется подстановкой в многочлен

$$x_i = \oplus_{i_1, \dots, i_k \in \{0,1\}} a_{i_1, \dots, i_k} x_{i_1}^{i_1} \dots x_{i_k}^{i_k}$$

Многочлен – заранее заданный параметр генератора.

Реализован в файле NFSR.java.

```
C:\Users\rost1\IdeaProjects\GPSCH1\out\artifacts\GPSCH1_jar>java -jar GPSCH1.jar
/g:nfsr /n:5 /i:3,107,24,0b100000000000001000000001,2312357,7,0b1000011,33,10,0
b101000101,7,0b110,0b001,0b111,0b000
Генератор создан
Начало генерирования
43360953410489126263661367656975
1872104025863439948258012635024
648697282432568781662468670211
127745515806399243493841715091471
142991416058404840163220561068032
Генерирование успешно завершено

C:\Users\rost1\IdeaProjects\GPSCH1\out\artifacts\GPSCH1_jar>
```

Рисунок 6 – пример запуска ГПСЧ на основе нелинейной комбинации РСЛОС

1.6 Вихрь Мерсена

Параметры:

w - разрядность,

r - позиция разделения,
 два положительных числа $q \leq p$,
 a, b, c - три w -разрядных неотрицательных
 числа, коэффициенты $0 \leq u, s, t, l \leq w$,
 p начальных значений X_0, X_{p-1} .

Если A, B - два w -разрядных числа, будет обозначать с помощью $(A|B)_w^r$ число, составленное из r младших бит числа B и $w - r$ старших бит числа A .

Очередное значение последовательности состояний вычисляется как:

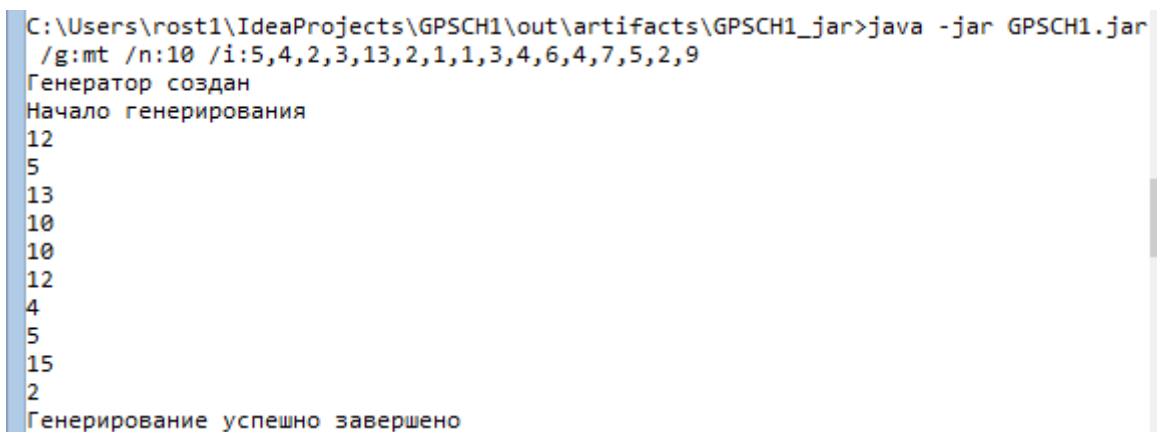
$$X_{i+p} = X_{i+q} \oplus ((X_i|X_{i+1})_w^r \gg 1) \oplus (a * ((X_i|X_{i+1})_w^r \bmod 2)) \bmod 2^w$$

На выход в качестве генерируемого числа выдаётся результат функции заделки $Z(X)$, определяемой следующим образом:

```

Z(X) :
  Y := X ⊕ (X ≫ u) (mod 2w)
  Y := Y ⊕ ((Y ≪ s) & b)
  Y := Y ⊕ ((Y ≪ t) & c)
  return Z(X) = Y ⊕ (Y ≫ l) (mod 2w)
  
```

Реализован в файле MT.java.



```

C:\Users\rosl1\IdeaProjects\GPSC1\out\artifacts\GPSC1_jar>java -jar GPSC1.jar
/g:mt /n:10 /i:5,4,2,3,13,2,1,1,3,4,6,4,7,5,2,9
Генератор создан
Начало генерирования
12
5
13
10
10
12
4
5
15
2
Генерирование успешно завершено
  
```

Рисунок 7 – пример запуска Вихря Мерсенна.

1.7 RC4

Генератор на основе поточного шифра RC4. Параметром является ключ — перестановка K_0, \dots, K_{255} чисел от 0 до 255. Состоянием генератора являются переменные i, j и массив S_0, \dots, S_{255} .

Инициализация происходит следующим образом:

$$j = 0, S_i = i$$

for *i* *in* 0..255:

$$j = j + S_i + K_i(\text{mod } 256); \text{swap}(S_i, S_j)$$

$$i = 0$$

$$t = 0$$

Генерация 8 бит:

$$i = i + 1 (\text{mod } 256)$$

$$j = j + S_i (\text{mod } 256)$$

$$\text{swap}(S_i, S_j)$$

$$X_t = S_{S_i + S_j}(\text{mod } 256)$$

$$t = t + 1$$

Реализован в файле RC4.java.

```
C:\Users\rosl1\IdeaProjects\GPSCH1\out\artifacts\GPSCH1_jar>java -jar GPSCH1.jar
/g:rc4 /n:10 /i:86,114,217,41,207,80,50,231,180,228,211,103,48,90,105,120,237,2
40,253,118,187,230,21,144,117,26,252,226,87,191,19,65,71,236,147,248,36,84,245,1
78,208,210,64,176,42,66,166,51,7,12,222,206,162,229,18,129,9,13,34,213,101,216,2
0,89,112,56,215,171,95,239,72,94,30,106,141,63,33,235,138,93,212,85,104,61,238,4
4,10,193,70,124,116,81,68,243,99,155,74,24,196,160,125,158,165,122,205,29,223,14
0,186,254,249,188,137,123,174,173,185,214,246,115,46,14,17,11,60,37,164,3,97,6,3
5,108,119,43,153,146,255,200,136,31,58,5,27,151,57,242,22,190,109,47,224,143,184
,232,227,149,163,96,110,142,82,111,49,203,69,98,126,38,113,28,15,195,55,67,40,15
9,88,152,100,194,102,52,250,218,1,167,169,0,83,189,179,247,177,181,168,75,198,92
,91,134,148,161,241,32,183,130,77,175,78,2,133,127,39,244,221,54,234,8,139,201,2
51,62,4,192,16,107,157,59,121,204,156,25,145,73,202,172,154,131,128,220,79,209,2
19,197,132,225,45,23,135,233,53,199,76,150,170,182,9
Генератор создан
Начало генерирования
360
509
380
274
203
94
419
66
289
292
Генерирование успешно завершено
```

Рисунок 8 – пример запуска ГПСЧ на основе RC4.

1.8 RSA

Генератор на основе RSA. Параметры: n — произведение двух простых чисел, e — степень, x_0 — начальное состояние. Генератор вырабатывает по w битов. Генерация происходит следующим образом: вычисляется $x_{i+1} = x_i^e \bmod n$, у полученного x_{i+1} выбираются w младших битов.

Реализован в файле RSA.java.

```
C:\Users\rost1\IdeaProjects\GPSC1\out\artifacts\GPSC1.jar>java -jar GPSC1.jar
/g:rsa /i:221,3,5,4,23 /n:5
Генератор создан
Начало генерирования
6
5
7
4
14
Генерирование успешно завершено
```

Рисунок 9 – пример запуска ГПСЧ на основе RSA.

1.9 Генератор Блюма-Блюма-Шуба

Генерируется последовательность битов. Параметры: n — произведение двух простых чисел, x_0 — начальное состояние. Для $x_i^2 \bmod n$ генерации бита вычисляется и берется его младший бит.

Реализован в файле BBS.java.

```
C:\Users\rost1\IdeaProjects\GPSC1\out\artifacts\GPSC1.jar>java -jar GPSC1.jar
/g:bbs /i:437,10,97 /n:5
Генератор создан
Начало генерирования
491
625
410
491
625
Генерирование успешно завершено
C:\Users\rost1\IdeaProjects\GPSC1\out\artifacts\GPSC1.jar>
```

Рисунок 10 – пример запуска ГПСЧ Блюма-Блюма-Шура

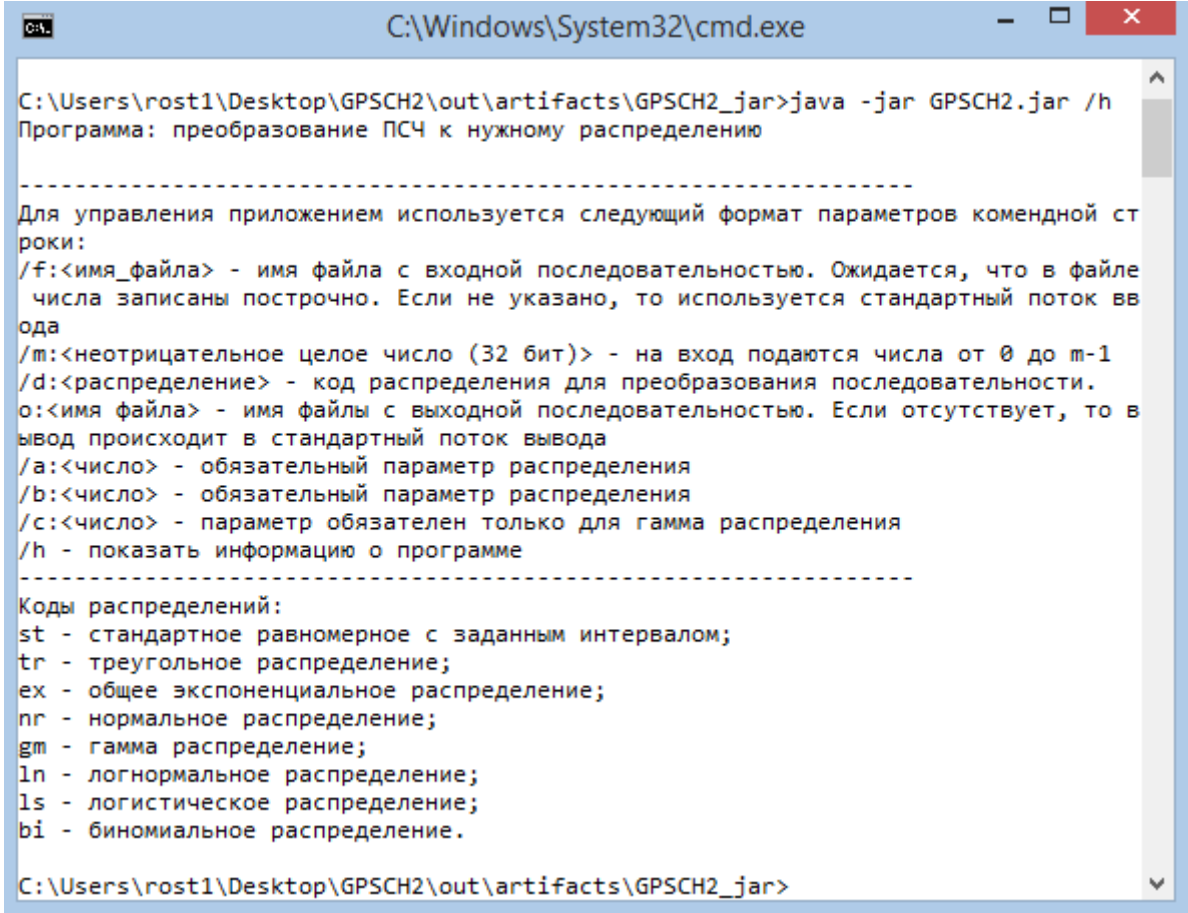
2 Приведение к распределению

Реализована программа преобразования последовательности ПСЧ к заданному распределению с заданными параметрами. Программа работает как фильтр, принимая последовательность на вход и выводя результат на

консольный выход (реализован ввод/вывод как с консоли, так и с файлов).

На входе должна быть последовательность целых чисел от 0 до $m - 1$ (m подается в программу как один из параметров при запуске)

Основная логика запуска и выдачи результатов реализована в файле Main.java из прилагаемого архива. Каждый распределение реализовано в виде отдельного Java-класса – наследника абстрактного класса Dimension, описанного в файле Main.java.



```
C:\Windows\System32\cmd.exe

C:\Users\rost1\Desktop\GPSCCH2\out\artifacts\GPSCCH2_jar>java -jar GPSCCH2.jar /h
Программа: преобразование ПСЧ к нужному распределению

-----
Для управления приложением используется следующий формат параметров комендной строки:
/f:<имя_файла> - имя файла с входной последовательностью. Ожидается, что в файле числа записаны построчно. Если не указано, то используется стандартный поток ввода
/m:<неотрицательное целое число (32 бит)> - на вход подаются числа от 0 до m-1
/d:<распределение> - код распределения для преобразования последовательности.
o:<имя файла> - имя файла с выходной последовательностью. Если отсутствует, то вывод происходит в стандартный поток вывода
/a:<число> - обязательный параметр распределения
/b:<число> - обязательный параметр распределения
/c:<число> - параметр обязателен только для гамма распределения
/h - показать информацию о программе
-----

Коды распределений:
st - стандартное равномерное с заданным интервалом;
tr - треугольное распределение;
ex - общее экспоненциальное распределение;
nr - нормальное распределение;
gm - гамма распределение;
ln - логнормальное распределение;
ls - логистическое распределение;
bi - биномиальное распределение.

C:\Users\rost1\Desktop\GPSCCH2\out\artifacts\GPSCCH2_jar>
```

Рисунок 11 – Справка о параметрах командной строки.

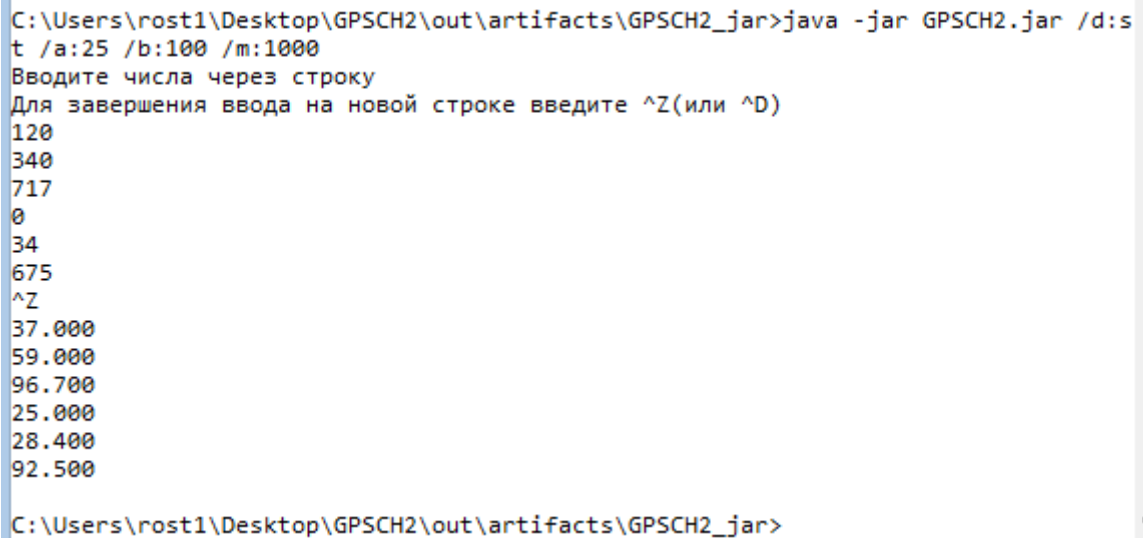
2.1 Равномерное распределение

Равномерное распределение $Unifrom(a; a + b)$, плотность вероятности $f(x) = 1/b$ для $x \in (a; a + b)$ и 0 иначе.

Введём обозначение преобразования последовательности для получения $Uniform(0; 1)$: $U(x, m) = x/m$.

Тогда для получения $Uniform(a; a + b)$ используем соотношение: $st(x, a, b, m) = a + U(x, m) * b$.

Программа преобразования реализована в файле St.java.



```
C:\Users\rost1\Desktop\GPSCH2\out\artifacts\GPSCH2_jar>java -jar GPSCH2.jar /d:s
t /a:25 /b:100 /m:1000
Вводите числа через строку
Для завершения ввода на новой строке введите ^Z(или ^D)
120
340
717
0
34
675
^Z
37.000
59.000
96.700
25.000
28.400
92.500
C:\Users\rost1\Desktop\GPSCH2\out\artifacts\GPSCH2_jar>
```

Рисунок 12 — Пример запуска преобразования к равномерному распределению.

2.2 Треугольное распределение

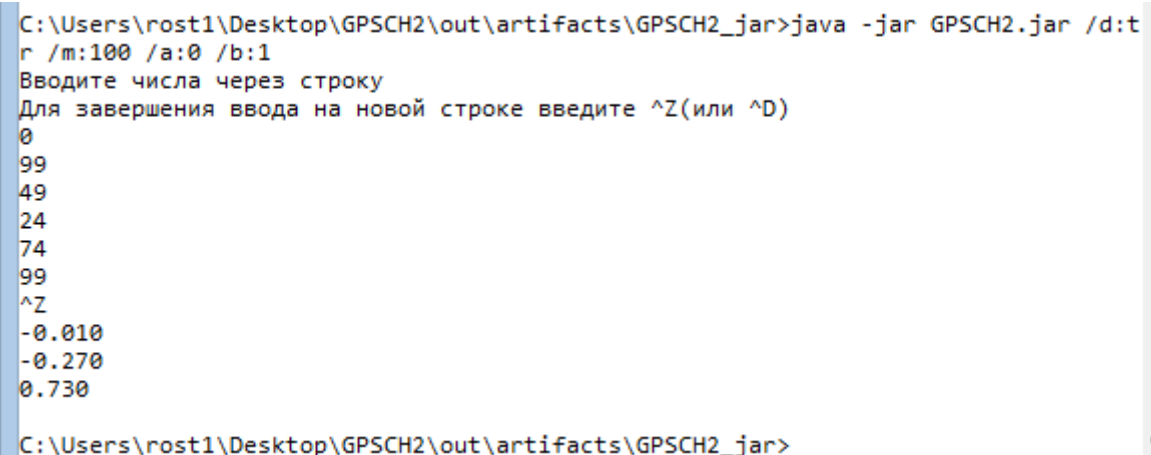
Функция распределения:

$$a - b \leq x < a + b : f(x) = \frac{b - |a - x|}{b^2}$$

Функция преобразования

$$tr(x_1, x_2, a, b, m) = a + b * (U(x_1, m) + U(x_2, m) - 1)$$

Программа преобразования реализована в файле Tr.java.



```
C:\Users\rost1\Desktop\GPSCH2\out\artifacts\GPSCH2_jar>java -jar GPSCH2.jar /d:t
r /m:100 /a:0 /b:1
Вводите числа через строку
Для завершения ввода на новой строке введите ^Z(или ^D)
0
99
49
24
74
99
^Z
-0.010
-0.270
0.730
C:\Users\rost1\Desktop\GPSCH2\out\artifacts\GPSCH2_jar>
```

Рисунок 13 — Пример запуска преобразования к треугольному распределению.

2.3 Экспоненциальное распределение

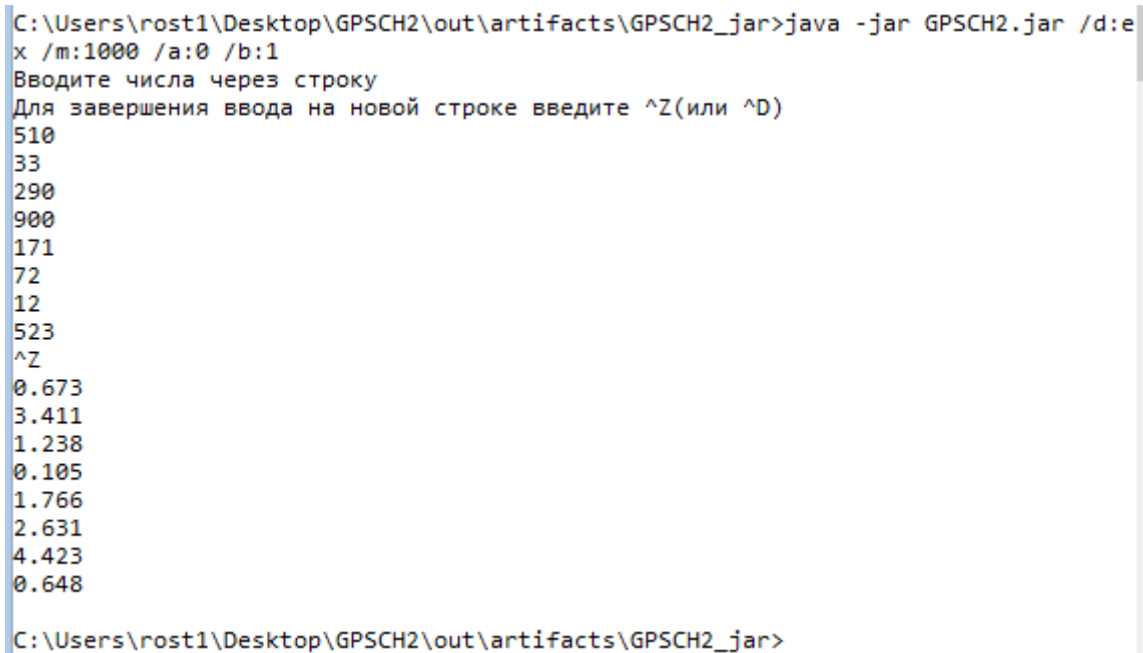
Функция распределения:

$$x \geq a, f(x) = \frac{1}{b} \exp\left(-\frac{x-a}{b}\right)$$

Функция преобразования:

$$ex(x, a, b, m) = a - b * \ln(U(x, m))$$

Программа преобразования реализована в файле Ex.java.



```
C:\Users\rost1\Desktop\GP SCH2\out\artifacts\GP SCH2_jar>java -jar GP SCH2.jar /d:e
x /m:1000 /a:0 /b:1
Вводите числа через строку
Для завершения ввода на новой строке введите ^Z(или ^D)
510
33
290
900
171
72
12
523
^Z
0.673
3.411
1.238
0.105
1.766
2.631
4.423
0.648
C:\Users\rost1\Desktop\GP SCH2\out\artifacts\GP SCH2_jar>
```

Рисунок 14 – пример запуска преобразования к экспоненциальному распределению.

2.4 Нормальное распределение

Функция распределения:

$$f(x) = \frac{1}{\sqrt{2\pi b}} \exp\left\{-\frac{1}{2b^2}(x - a)^2\right\}$$

Преобразование:

$$\begin{aligned} y_1, y_2 &= nr(x_1, x_2, a, b) : \\ y_1 &= a + b * \sqrt{-2\ln(1 - U(x_1))} \cos(2\pi U(x_2)) \\ y_2 &= a + b * \sqrt{-2\ln(1 - U(x_1))} \sin(2\pi U(x_2)) \end{aligned}$$

Программа преобразования реализована в файле Nr.java.

```

C:\Users\rost1\Desktop\GPSCH2\out\artifacts\GPSCH2_jar>java -jar GPSCH2.jar /d:n
r /m:1000 /a:0 /b:1
Вводите числа через строку
Для завершения ввода на новой строке введите ^Z(или ^D)
510
33
290
900
171
72
124
523
^Z
1.169
0.246
0.670
-0.486
0.551
0.268
-0.509
-0.074

```

Рисунок 15 – пример запуска преобразования к нормальному распределению.

2.5 Логнормальное распределение

Случайная величина, логарифм которой нормально распределен, имеет логнормальное распределение.

Функция распределения:

$$f(x) = 1/(\sqrt{2\pi}\{\frac{x-a}{b}\})\exp\{-\frac{1}{2}(\frac{x-a}{b})^2\} \text{ при } x \geq a$$

Преобразование:

$$\begin{aligned}
 y_1, y_2 &= \text{lognorm}(x_1, x_2, a, b) : \\
 z_1, z_2 &= \text{norm}(x_1, x_2, 0, 1, m) \\
 y_1 &= a + \exp(b - z_1) \\
 y_2 &= a + \exp(b - z_2)
 \end{aligned}$$

Программа преобразования реализована в файле Ln.java.

```

C:\Users\rost1\Desktop\GPSCH2\out\artifacts\GPSCH2_jar>java -jar GPSCH2.jar /d:1
n /m:1000 /a:-5 /b:2
Вводите числа через строку
Для завершения ввода на новой строке введите ^Z(или ^D)
110
332
250
132
171
725
1
999
^Z
4.373
-0.146
-0.573
-0.777
3.132
8.530
2.066
2.391

```

Рисунок 16 – пример запуска преобразования к нормальному распределению.

2.6 Логистическое распределение

Для получения логистического распределения используется преобразование:

$$\begin{aligned}
 y &= \text{logistic}(x, a, b, m) : \\
 u &= U(x, m) \\
 y &= a + b * \ln\left(\frac{u}{1-u}\right)
 \end{aligned}$$

Программа преобразования реализована в файле Ls.java.

```

C:\Users\rost1\Desktop\GPSCH2\out\artifacts\GPSCH2_jar>java -jar GPSCH2.jar /d:1
s /m:1000 /a:0 /b:1
Вводите числа через строку
Для завершения ввода на новой строке введите ^Z(или ^D)
510
33
290
900
171
72
124
523
^Z
0.040
-3.378
-0.895
2.197
-1.579
-2.556
-1.955
0.092

```

Рисунок 17 – пример запуска преобразования к логистическому распределению.

2.7 Биномиальное распределение

Дискретное распределение.

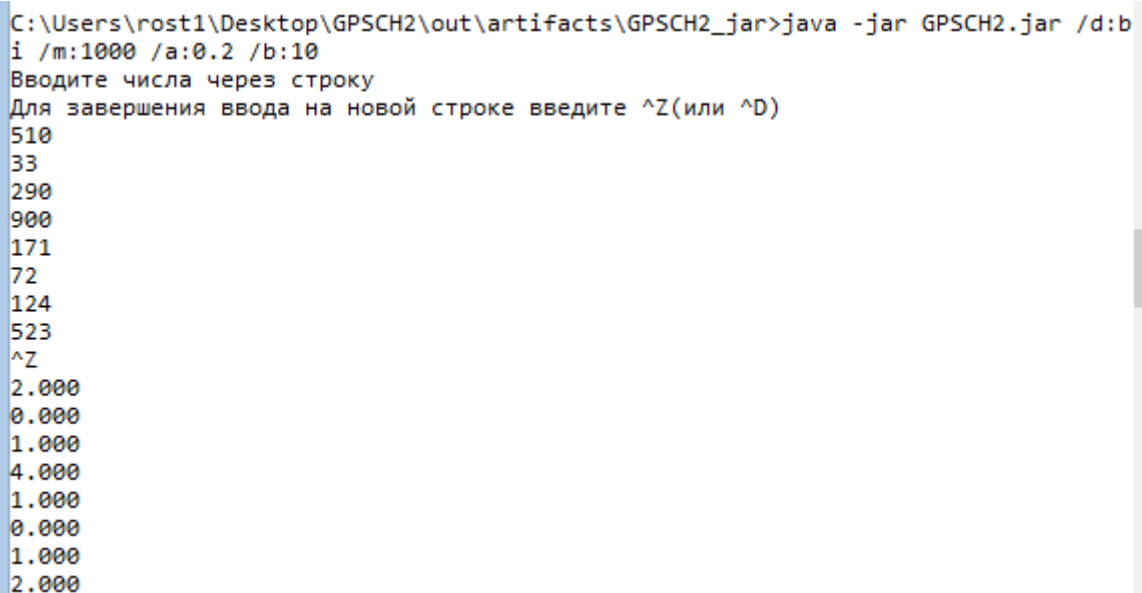
Преобразование:

```

 $y = binominal(x, a, b, m) :$ 
     $u = U(x)$ 
     $s = 0$ 
     $k = 0$ 
loopstart :
     $s = s + C_b^k a^k (1 - a)^{b-k}$ 
    if  $s > u$  :
         $y = k$ 
        Завершить
    if  $k < b - 1$  :
         $k = k + 1$ 
        перейти к loopstart
 $y = b$ 

```

Программа преобразования реализована в файле Vi.java.



```

C:\Users\rost1\Desktop\GPSCCH2\out\artifacts\GPSCCH2_jar>java -jar GPSCCH2.jar /d:b
i /m:1000 /a:0.2 /b:10
Вводите числа через строку
Для завершения ввода на новой строке введите ^Z(или ^D)
510
33
290
900
171
72
124
523
^Z
2.000
0.000
1.000
4.000
1.000
0.000
1.000
2.000

```

Рисунок 18 – пример запуска преобразования к биномиальному распределению.

2.8 Гамма распределение

У распределения три параметра: a , b , c . Параметр c имеет ограничение: может быть или целым, или полуцелым числом.

Преобразование:

При $c = k$ при $k \in \mathbb{Z}, k > 0$

$$y = gamma_k(x_1, \dots, x_c, a, b, c, m) = a - b * \ln\{[1 - U(x_1, m)] * \dots * [1 - U(x_m)]\}$$

При $c = k + 0.5$ при $k \in \mathbb{Z}, k \geq 0$

$$y_1, y_2 = \text{gamma}_{k+0.5}(x_1, \dots, x_k, x_{k+1}, \dots, x_{2k}, x_{2k+1}, x_{2k+2}, a, b, m) :$$

$$z_1, z_2 = \text{norm}(x_1, x_2, 0, 1, m)$$

$$y_1 = a + b * \left(\frac{z_1^2}{2} - \ln\{[1 - U(x_3, m)] * \dots * [1 - U(x_{k+2}, m)]\} \right)$$

$$y_2 = a + b * \left(\frac{z_2^2}{2} - \ln\{[1 - U(x_{k+3}, m)] * \dots * [1 - U(x_{k+2}, m)]\} \right)$$

Программа преобразования реализована в файле Gm.java.

```
C:\Users\rost1\Desktop\GPSCH2\out\artifacts\GPSCH2_jar>java -jar GPSCH2.jar /d:gm /m:1000 /a:0 /b:0.5 /c:2
Вводите числа через строку
Для завершения ввода на новой строке введите ^Z(или ^D)
510
33
290
900
171
72
124
523
^Z
0.373
1.323
0.131
0.436
```

Рисунок 19 – пример запуска преобразования к биномиальному распределению с целым c .

```
C:\Users\rost1\Desktop\GPSCH2\out\artifacts\GPSCH2_jar>java -jar GPSCH2.jar /d:gm /m:1000 /a:0 /b:1 /c:1.5
Вводите числа через строку
Для завершения ввода на новой строке введите ^Z(или ^D)
99
59
56
59
42
86
5
94
3
92
56
69
^Z
0.148
0.074
0.037
0.110
0.060
0.072
```

Рисунок 20 – пример запуска преобразования к биномиальному распределению с полуцелым c .