

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ 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

Дусалиева Тахира Ахатовича

фамилия, имя, отчество

Научный руководитель

Ст. преподаватель

подпись, дата

И.И. Слеповичев

Саратов 2023

Задание 1. Генерация псевдослучайных чисел.

Описание задания: Создать программу для генерации псевдослучайных величин следующими алгоритмами:

- a. Линейный конгруэнтный метод;
- b. Аддитивный метод;
- c. Пятипараметрический метод;
- d. Регистр сдвига с обратной связью (РСЛОС);
- e. Нелинейная комбинация РСЛОС;
- f. Вихрь Мерсенна;
- g. RC4;
- h. ГПСЧ на основе RSA;
- i. Алгоритм Блюма-Блюма-Шуба.

Название программы: **prng.exe**.

Входные параметры алгоритмов передаются программе через строку параметров. Для управления программой сформулирован следующий формат параметров командной строки.

/g:<Метод> — параметр указывает на метод генерации ПСЧ, при этом в параметр должен передаваться один из следующих методов:

- lc – линейный конгруэнтный метод;
- add – аддитивный метод;
- 5p – пятипараметрический метод;
- lfsr – регистр сдвига с обратной связью (РСЛОС);
- nfsr – нелинейная комбинация РСЛОС;
- mt – вихрь Мерсенна;
- rc4 – RC4;
- rsa – ГПСЧ на основе RSA;
- bbs – алгоритм Блюма-Блюма-Шуба.

/i:<Параметры> — вектор всех необходимых параметров для алгоритмов. Элементы вектора параметров разделяются запятой. Порядок элементов вектора для каждого из методов описан в таблице 1.

Таблица 1 — Порядок элементов вектора параметров.

Метод	Описание параметров
lc	Модуль, множитель, приращение, начальное значение
add	Модуль, младший индекс, старший индекс, последовательность начальных значений
5p	p, q1, q2, q3, w, начальное значение
lfsr	Двоичное представление вектора коэффициентов, начальное значение регистра
nfsr	В алгоритме использовать три РСЛОС R1, R2, R3, скомбинированных функцией $R1 \wedge R2 + R2 \wedge R3 + R3$. Параметры – двоичное представление векторов коэффициентов для R1, R2, R3, w, x1, x2, x3. w – длина слова, x1, x2, x3 – десятичное представление начальных состояний регистров R1, R2, R3.
mt	Модуль, начальное значение x
rc4	256 начальных значений
rsa	Модуль n, число e, w, начальное значение x. e удовлетворяет условиям: $1 < e < (p-1)(q-1)$, $\text{НОД}(e, (p-1)(q-1)) = 1$, где $p \cdot q = n$. x из интервала $[1, n]$ w – длина слова.
bbs	Начальное значение x (взаимно простое с n). При генерации использовать параметры: $p = 127$, $q = 131$, $n = p \cdot q = 16637$

/n:<Кол-во> — количество генерируемых ПСЧ, по умолчанию значение равняется 10 000.

/m:<Модуль> — максимальное значение генерируемых ПСЧ, по умолчанию значение равняется 1024 и ПСЧ генерируются в отрезке $[0, 1023]$.

/f:<Путь> — путь для выходного файла, по умолчанию файл сохраняется в каталог с программой под названием «rnd.dat».

/h — информация о допустимых параметрах командной строки программы.

Выходные значения записываются в файл, указанный в параметре **/f** запуска программы.

Алгоритм 1. Линейный конгруэнтный метод.

Описание алгоритма

Одним из простых и популярных методов сейчас является *линейный конгруэнтный метод* (ЛКМ), предложенный Д.Г. Лехмером в 1949 году. В его основе лежит выбор четырех ключевых чисел:

- $m > 0$, модуль;
- $0 \leq a \leq m$, множитель;
- $0 \leq c \leq m$, приращение (инкремент);
- $0 \leq X_0 \leq m$, начальное значение.

Последовательность ПСЧ, вычисляется по формуле:

$$X_{n+1} = (aX_n + c) \bmod m, n \geq 1$$

Теорема 3.1. Линейная конгруэнтная последовательность, определенная числами m , a , c и X_0 , имеет период длиной m тогда и только тогда, когда:

- 1) числа c и m взаимно простые;
- 2) $a - 1$ кратно p для некоторого простого p , являющегося делителем m ;
- 3) $a - 1$ кратно 4, если m кратно 4.

Доказательство приведено в [1, Том 2. п. 3.2.1.2].

Параметры запуска программы

prng.exe /g:lc /i:m,a,c,X₀.

Исходный текст программы

```
import math

"""
Реализация линейно-конгруэнтного метода генерации псевдослучайных чисел
"""

def linear_rand_method(m : int, modulus : int, a : int, c : int, seed,
size=10000):
    sr = seed
    if size == 1:
        return math.ceil(math.fmod(a * math.ceil(sr) + c, modulus))
    r = [0 for _ in range(size + 1)]
```

```

r[0] = math.ceil(sr)
for i in range(1, size + 1):
    r[i] = math.ceil(math.fmod((a * r[i - 1] + c), modulus))
return [r[i] % m for i in range(1, size + 1)]

```

Пример работы программы

Введём следующие параметры для запуска программы: `prng.exe /g:lc /i:2147483647,48271,0,350 /n:10000 /m:1024 /f:rnd-lc.dat`.

```

1 898, 793, 646, 542, 720, 540, 882, 569, 758, 678, 505, 646, 661, 97, 989, 205, 320, 440,
640, 596, 31, 819, 946, 982, 41, 40, 901, 452, 55, 336, 797, 617, 85, 120, 794, 774, 105,
68, 873, 62, 1001, 875, 923, 613, 780, 672, 243, 803, 244, 473, 107, 263, 909, 740, 322,
767, 721, 762, 600, 33, 77, 503, 838, 791, 759, 704, 201, 815, 760, 469, 88, 859, 14, 877,
774, 823, 253, 360, 260, 125, 397, 610, 25, 748, 585, 851, 36, 207, 784, 981, 17, 11, 217,
347, 273, 460, 764, 53, 546, 343, 517, 14, 537, 626, 227, 198, 632, 387, 305, 448, 120,
187, 14, 674, 42, 819, 637, 810, 293, 306, 922, 88, 1000, 841, 545, 304, 447, 719, 938,
263, 185, 792, 127, 503, 612, 385, 569, 39, 650, 602, 658, 349, 559, 654, 790, 643, 382,
263, 542, 964, 331, 488, 1009, 737, 480, 969, 703, 702, 44, 124, 155, 669, 150, 532, 978,
626, 186, 211, 580, 531, 51, 846, 552, 661, 758, 756, 806, 793, 996, 845, 177, 516, 885,
249, 1008, 55, 801, 661, 105, 24, 671, 269, 526, 422, 235, 776, 85, 899, 385, 28, 737,
596, 650, 388, 663, 916, 7, 434, 12, 722, 921, 714, 687, 603, 371, 490, 59, 430, 778, 194,
788, 10, 368, 118, 269, 570, 498, 483, 82, 880, 498, 595, 427, 112, 397, 620, 851, 748,
22, 570, 747, 271, 660, 927, 524, 966, 618, 96, 419, 494, 356, 873, 831, 902, 96, 1015,
162, 20, 1003, 381, 850, 441, 481, 517, 165, 702, 609, 640, 228, 53, 735, 140, 324, 429,
547, 470, 172, 895, 906, 132, 830, 716, 105, 462, 756, 150, 816, 824, 113, 753, 474, 908,
332, 706, 102, 12, 383, 42, 446, 194, 450, 133, 602, 157, 542, 278, 1022, 626, 630, 6,
756, 90, 465, 170, 151, 469, 936, 252, 205, 228, 732, 290, 50, 332, 548, 114, 901, 708,
45, 614, 936, 826, 343, 986, 981, 961, 881, 406, 875, 313, 817, 472, 994, 327, 568, 226,
5, 610, 350, 297, 1005, 628, 709, 282, 872, 239, 424, 85, 50, 469, 498, 197, 667, 761,
1008, 623, 889, 929, 754, 854, 13, 694, 781, 221, 207, 678, 314, 357, 928, 522, 120, 481,
350, 348, 714, 827, 790, 170, 161, 926, 375, 61, 973, 133, 926, 987, 620, 193, 200, 967,
223, 532, 839, 465, 110, 214, 884, 920, 29, 52, 486, 130, 827, 830, 319, 582, 142, 838,
441, 505, 202, 446, 560, 26, 544, 4, 984, 224, 298, 8, 262, 157, 952, 914, 585, 616, 318,
98, 336, 999, 803, 208, 553, 202, 952, 56, 936, 991, 231, 1002, 915, 674, 84, 3, 835, 58,
94, 216, 929, 892, 613, 363, 733, 300, 647, 906, 196, 465, 923, 824, 952, 307, 614, 424,
888, 787, 307, 457, 829, 393, 793, 433, 262, 120, 367, 818, 960, 532, 293, 802, 464, 151,
951, 861, 662, 143, 470, 531, 117, 717, 854, 641, 15, 860, 984, 639, 435, 109, 1008, 577,
963, 562, 812, 9, 581, 159, 1009, 53, 968, 499, 395, 919, 792, 537, 14, 493, 612, 383,
801, 872, 893, 137, 56, 369, 812, 946, 101, 101, 320, 431, 324, 431, 156, 676, 1022, 294,
463, 221, 295, 198, 644, 982, 576, 39, 968, 403, 494, 789, 150, 414, 245, 90, 852, 863,
427, 820, 305, 845, 786, 945, 259, 342, 295, 740, 14, 807, 446, 875, 601, 927, 14, 166,
892, 903, 482, 601, 443, 606, 936, 658, 409, 728, 674, 109, 443, 334, 193, 425, 523, 621,
976, 324, 43, 243, 125, 951, 599, 461, 410, 23, 300, 726, 250, 1020, 209, 1017, 642, 595,
133, 902, 908, 351, 31, 487, 24, 660, 305, 431, 578, 980, 436, 922, 5, 444, 204, 473, 668,
339, 730, 428, 320, 174, 303, 203, 382, 629, 480, 433, 967, 395, 995, 82, 409, 332, 939,
304, 891, 827, 268, 456, 455, 704, 124, 87, 931, 636, 946, 395, 345, 695, 615, 338, 901,
274 449 668 542 932 538 733 381 116 543 355 928 208 401 622 137 234 169

```

Алгоритм 2. Аддитивный метод.

Описание алгоритма

Каждое следующее значение вычисляется по рекуррентной формуле вычисления последовательности Фибоначчи с запаздыванием:

$$X_{n+1} = (X_{n-k} + X_{n-j}) \bmod m, \quad j > k \geq 1,$$

где k, j — запаздывания.

Параметры запуска программы

prng.exe /g:add /i:m,k,j,x₀, x₁, x₂, ..., x_j

Исходный текст программы

```
import math

"""
Реализация аддитивного метода генерации псевдослучайных чисел
"""

def add_rand_method(m : int, modulus: int, k : int, l : int, x0 :
list[int], size = 10000):
    n = len(x0) - 1
    if size == 1:
        return math.ceil(math.fmod((x0[n - k] + x0[n - 1]), modulus))
    r = x0.copy()
    r.extend([0 for _ in range(0, size)])
    for _ in range(0, size):
        r[n + 1] = math.ceil(math.fmod((r[n - k] + r[n - 1]),
modulus))
        n += 1
    return [r[i] % m for i in range(1, len(r))]
```

Пример работы программы

Введём следующие параметры для запуска программы: prng.exe /g:add /i:100,24,55,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26, 27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,5 3,54,55 /n:10000 /m:1024 /f:rnd-add.dat.

md-add.dat	
1	31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 56, 59, 62, 65, 68, 71, 74, 77, 80, 83, 86, 89, 92, 95, 98, 1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 6, 10, 14, 18, 22, 26, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 75, 81, 87, 93, 99, 5, 61, 69, 77, 85, 93, 1, 9, 17, 25, 33, 41, 49, 57, 65, 73, 81, 89, 97, 5, 88, 97, 6, 15, 24, 33, 67, 79, 91, 3, 15, 27, 14, 27, 40, 53, 66, 79, 92, 5, 18, 31, 44, 57, 70, 58, 72, 86, 0, 14, 28, 42, 60, 78, 96, 14, 32, 75, 96, 17, 38, 59, 80, 1, 22, 43, 64, 85, 6, 27, 23, 45, 67, 89, 11, 33, 30, 57, 84, 11, 38, 65, 42, 75, 8, 41, 74, 7, 15, 49, 83, 17, 51, 85, 19, 28, 63, 98, 33, 68, 3, 88, 29, 70, 11, 52, 93, 84, 35, 86, 37, 88, 39, 90, 45, 0, 55, 10, 65, 20, 50, 6, 62, 18, 74, 30, 11, 74, 37, 0, 63, 26, 14, 92, 70, 48, 26, 4, 32, 20, 8, 96, 84, 72, 35, 99, 89, 79, 69, 59, 49, 39, 37, 35, 33, 31, 29, 2, 21, 40, 59, 78, 97, 16, 55, 94, 33, 72, 11, 25, 44, 89, 34, 79, 24, 69, 89, 43, 97, 51, 5, 59, 13, 95, 77, 59, 41, 23, 30, 47, 64, 81, 98, 15, 57, 64, 97, 30, 63, 96, 4, 88, 32, 76, 20, 64, 8, 52, 32, 12, 92, 72, 52, 32, 68, 4, 40, 76, 12, 73, 19, 91, 63, 35, 7, 29, 32, 21, 10, 99, 88, 77, 41, 75, 9, 43, 77, 11, 45, 63, 81, 99, 17, 35, 3, 66, 55, 44, 33, 22, 86, 96, 18, 40, 62, 84, 81, 29, 7, 85, 63, 41, 19, 97, 95, 93, 91, 89, 87, 35, 34, 59, 84, 9, 34, 59, 15, 9, 3, 97, 91, 10, 61, 28, 95, 62, 29, 96, 38, 70, 2, 34, 66, 98, 80, 97, 40, 83, 26, 69, 62, 81, 64, 47, 30, 13, 96, 57, 46, 35, 24, 13, 77, 67, 77, 87, 97, 7, 17, 77, 92, 33, 74, 15, 56, 97, 15, 23, 31, 39, 47, 55, 72, 55, 38, 21, 4, 87, 28, 5, 82, 59, 36, 13, 15, 62, 35, 8, 81, 54, 77, 12, 63, 14, 65, 16, 17, 53, 19, 85, 51, 17, 83, 85, 51, 17, 83, 49, 90, 82, 39, 22, 5, 88, 71, 54, 4, 96, 88, 80, 72, 14, 68, 42, 16, 90, 64, 38, 57, 6, 55, 4, 53, 77, 10, 44, 4, 64, 24, 84, 69, 66, 31, 96, 61, 26, 91, 80, 5, 30, 55, 80, 55, 10, 25, 40, 55, 70, 60, 95, 95, 21, 47, 73, 74, 51, 5, 53, 1, 49, 97, 45, 84, 1, 18, 35, 52, 69, 78, 67, 56, 45, 34, 98, 52, 1, 76, 51, 26, 51, 61, 49, 57, 65, 73, 81, 14, 50, 32, 14, 96, 78, 60, 58, 72, 86, 0, 14, 53, 62, 26, 16, 6, 96, 11, 56, 44, 78, 12, 46, 55, 65, 55, 85, 15, 45, 75, 5, 42, 73, 4, 35, 66, 22, 40, 93, 72, 51, 30, 9, 8, 45, 54, 63, 72, 6, 26, 4, 42, 80, 18, 56, 19, 92, 5, 18, 31, 44, 82, 98, 65, 58, 51, 44, 62, 70, 71, 70, 69, 68, 17, 82, 48, 20, 92, 64, 11, 84, 47, 90, 33, 76, 19, 87, 40, 38, 62, 86, 10, 84, 10, 64, 42, 20, 98, 26, 90, 93, 74, 55, 36, 17, 10, 51, 32, 13, 94, 75, 6, 32, 43, 80, 17, 54, 66, 8, 29, 0, 71, 42, 88, 60, 64, 44, 24, 4, 34, 92, 99, 52, 5, 58, 86, 90, 79, 33, 13, 93, 73, 53, 48, 67, 62, 57, 52, 72, 70, 28, 86, 44, 2, 60, 82, 92, 26, 60, 94, 3, 0, 30, 65, 26, 87, 48, 59, 80, 10, 42, 74, 6, 38, 78, 57, 86, 15, 44, 48, 42, 56, 70, 84, 98, 37, 92, 29, 17, 31, 45, 34, 49, 59, 43, 55, 67, 79, 91, 26, 24, 48, 72, 96, 20, 12, 84, 56, 28, 0, 97, 74, 21, 43, 91, 39, 37, 49, 89, 8, 81, 54, 27, 50, 6, 34, 90, 46, 2, 58, 90, 41, 42, 43, 44, 45, 16, 77, 13, 75, 37, 74, 41, 18, 25, 12, 99, 61, 99, 65, 77, 45, 13, 81, 49, 16, 65, 90, 15, 40, 65, 28, 61, 69, 3, 37, 71, 15, 39, 68, 3, 38, 98, 48, 54, 85, 26, 67, 8, 99, 22, 99, 80, 61, 42, 23, 18, 2, 11, 46, 81, 16, 31, 16, 81, 78, 75, 72, 89, 72, 10, 38, 66, 69, 98, 87, 76, 25, 74, 23, 72 34 67 1 61 21 81 59 77 50 81 12 43 4 11 78 41 4 67 46 41 61 51
Normal t length : 39 005 lines : 1 Ln : 1 Col : 1 Pos : 1 Windows (CR LF) UTF-8 INS	

Алгоритм 3. Пятипараметрический метод.

Описание алгоритма

Данный метод является частным случаем РСЛОС, использует характеристический многочлен из 5 членов и позволяет генерировать последовательности w -битовых двоичных целых чисел в соответствии со следующей рекуррентной формулой:

$$X_{n+p} = X_{n+q_1} + X_{n+q_2} + X_{n+q_3} + X_n, n = 1, 2, 3, \dots$$

Параметры (p, q_1, q_2, q_3, w) и X_p начальный вектор состоящий из $X_0 \dots X_{p-1}$ битов.

Параметры запуска программы

`prng.exe /g:5p /i:p,q1,q2,q3,w,Xp`

Исходный текст программы

```
"""
Реализация пятипараметрического метода генерации псевдослучайных чисел
"""

def five_P_rand_method(m : int, p : int, q1 : int, q2 : int, q3 :
int, w : int, seed : str, size=10000):
    sr = seed
    r = [0 for _ in range(size)]
    for i in range(size):
        for _ in range(w):
            xor = ((sr >> q1) ^ (sr >> q2) ^ (sr >> q3) ^ (sr & 1)) &
1
            r[i] <<= 1
            r[i] ^= sr & 1
            sr >>= 1
            sr |= xor << (p - 2)
        r[i] %= m
    return r
```

Пример работы программы

Введём следующие параметры для запуска программы: `prng.exe /g:5p /i:89,20,40,69,1024,615930009644690137449363111 /n:10000 /f:rnd_5p.dat /m:1024`.

md_5p.dat	
1	832, 339, 8, 678, 706, 420, 686, 464, 940, 561, 86, 227, 254, 630, 728, 208, 991, 761, 197, 473, 301, 974, 328, 259, 943, 835, 255, 528, 621, 730, 626, 880, 153, 832, 56, 833, 686, 7, 372, 836, 108, 664, 660, 71, 234, 637, 519, 921, 634, 920, 162, 844, 661, 740, 556, 427, 2, 734, 541, 768, 693, 244, 54, 701, 923, 204, 393, 866, 501, 575, 524, 304, 905, 203, 1018, 962, 799, 758, 292, 294, 147, 256, 93, 10, 445, 887, 723, 872, 894, 858, 748, 717, 910, 708, 314, 662, 737, 102, 1014, 175, 780, 505, 542, 855, 475, 10, 1, 826, 387, 128, 111, 372, 437, 907, 280, 538, 1001, 593, 513, 368, 229, 49, 641, 946, 791, 41, 3, 933, 678, 411, 287, 770, 429, 675, 320, 995, 218, 868, 400, 792, 54, 142, 370, 579, 471, 659, 375, 815, 516, 486, 321, 752, 709, 573, 305, 136, 409, 586, 411, 911, 693, 61, 609, 347, 923, 832, 728, 987, 527, 137, 623, 544, 241, 274, 138, 262, 367, 128, 767, 195, 37, 841, 259, 610, 115, 1005, 74, 103, 90, 720, 467, 256, 978, 560, 114, 485, 34, 642, 157, 471, 33, 413, 531, 792, 686, 356, 108, 205, 83, 352, 531, 202, 124, 417, 586, 214, 649, 498, 761, 241, 937, 316, 675, 800, 402, 678, 937, 799, 1008, 359, 936, 837, 585, 743, 580, 309, 935, 160, 845, 137, 543, 385, 984, 369, 1003, 427, 381, 740, 355, 380, 85, 819, 58, 205, 275, 781, 169, 253, 841, 981, 559, 73, 841, 728, 768, 525, 128, 69, 756, 244, 899, 2, 1009, 721, 200, 891, 862, 846, 622, 416, 636, 15, 726, 739, 803, 466, 497, 652, 322, 830, 377, 416, 168, 965, 431, 677, 477, 629, 605, 522, 636, 252, 436, 283, 732, 428, 648, 542, 782, 177, 305, 769, 814, 521, 856, 892, 482, 981, 476, 333, 823, 936, 192, 682, 603, 315, 825, 379, 226, 375, 670, 789, 554, 135, 525, 811, 99, 534, 951, 661, 241, 468, 973, 901, 952, 514, 328, 840, 375, 621, 119, 466, 376, 627, 82, 241, 90, 56, 733, 592, 594, 513, 370, 609, 573, 232, 201, 560, 806, 803, 166, 713, 756, 197, 320, 831, 980, 795, 511, 582, 773, 1020, 820, 618, 361, 266, 91, 254, 229, 730, 452, 987, 294, 633, 334, 487, 654, 741, 175, 450, 174, 448, 53, 691, 307, 629, 404, 32, 121, 529, 356, 194, 1020, 932, 102, 1014, 180, 481, 275, 949, 588, 643, 929, 765, 507, 695, 770, 111, 838, 153, 644, 1015, 240, 388, 868, 634, 546, 234, 727, 862, 337, 285, 755, 803, 1005, 764, 319, 194, 818, 184, 246, 41, 478, 27, 949, 764, 118, 540, 1022, 584, 967, 814, 523, 508, 198, 571, 382, 325, 602, 949, 957, 400, 211, 887, 106, 57, 291, 374, 424, 242, 34, 43, 766, 330, 173, 117, 641, 689, 168, 35, 194, 269, 353, 33, 516, 398, 258, 393, 640, 1020, 261, 865, 411, 43, 545, 119, 920, 446, 693, 753, 501, 958, 778, 707, 463, 133, 225, 397, 183, 188, 620, 759, 1012, 228, 290, 242, 1000, 15, 276, 752, 843, 956, 680, 897, 890, 789, 895, 801, 963, 753, 837, 162, 650, 149, 211, 791, 807, 538, 983, 1, 217, 969, 269, 888, 524, 112, 316, 273, 966, 502, 713, 104, 862, 472, 756, 803, 45, 151, 872, 335, 422, 450, 123, 649, 0, 141, 688, 967, 840, 715, 458, 540, 877, 356, 189, 641, 995, 293, 102, 1011, 188, 10, 888, 997, 66, 165, 260, 441, 345, 447, 441, 916, 18, 984, 194, 467, 288, 865, 639, 118, 478, 565, 450, 118, 943, 905, 262, 23, 192, 946, 738, 483, 468, 63, 607, 1014, 1007, 322, 885, 938, 518, 382, 575, 525, 71, 180, 793, 636, 680, 810, 736, 184, 125, 631, 66, 401, 851, 1, 985, 910, 513, 131, 626, 789, 631, 349, 101, 301, 584, 223, 514, 1006, 410, 878, 403, 765, 408, 468, 213, 118, 325, 405, 302, 82, 014, 582, 30, 404, 775, 320, 10, 003
Normal t length: 49 215 lines: 1 Ln: 1 Col: 1 Pos: 1 Windows (CR LF) UTF-8 INS	

Алгоритм 4. Регистр сдвига с обратной связью (РСЛОС).

Описание алгоритма

Следующий класс ГПСЧ основан на идее преобразования двоичного представления некоторого числа. Такие генераторы имеют некоторые преимущества, как, например, скорость генерации таких чисел, хорошие статистические свойства ПСЧ, а также возможность простой реализации на аппаратном уровне.

Для натурального числа p и двоичного представления a_p , состоящий из a_1, a_2, \dots, a_{p-1} битов, принимающих значения 0 или 1, определяют рекуррентную формулу

$$X_{n+p} = a_{p-1}X_{n+p-1} + a_{p-2}X_{n+p-2} + \dots + a_1X_{n+1} + X_n, \quad (3.7)$$

Как видно из формулы, для РСЛОС функция обратной связи является линейной булевой функцией от состояний всех или некоторых битов регистра.

Одна итерация алгоритма, генерирующего последовательность, состоит из следующих шагов:

1. Содержимое ячейки $p - 1$ формирует очередной бит ПСП битов.
2. Содержимое ячейки 0 определяется значением функции обратной связи, являющейся линейной булевой функцией с коэффициентами a_1, a_2, \dots, a_{p-1} . Его вычисляют по формуле 3.7.
3. Содержимое каждого i -го бита перемещается в $(i + 1)$ -й, $0 \leq i < p - 1$.
4. В ячейку 0 записывается новое содержимое, вычисленное на шаге 2.

Параметры запуска программы

prng.exe /g:lfsr /i: a_p, X_p

Исходный текст программы

```
"""
Реализация РСЛОС метода генерации псевдослучайных чисел
"""

def excep(R : str):
    for s in R:
        if s != '0':
            if s != '1':
                raise Exception

def lfsr_rand_method(m : int, c_vec : str, seed : str, size=10000):
    excep(c_vec)
    excep(seed)
    sr = int(seed, 2)
    xor = 0
    r = [0 for _ in range(size)]
    c = [i for i in range(len(c_vec) - 1, -1, -1) if c_vec[len(c_vec)
- 1 - i] != '0']
    for i in range(size):
        current = 0
        for _ in range(2 ** (c[0] - 1)):
            for ch in c:
                xor ^= (sr >> ch)
            xor ^= sr & 1
            current <=< 1
            current ^= (sr & 1)
            sr >>= 1
            sr |= (xor & 1) << (c[0] - 1)
            xor = 0
        r[i] = current % m
    return r
```

Пример работы программы

Введём следующие параметры для запуска программы: `prng.exe /g:lfsr /i:1000001010011,1110101001001 /n:10000 /f:rnd_lfsr.dat /m:1024`.


```
md_lfsr.dat
1 262, 959, 138, 358, 368, 863, 174, 896, 422, 238, 245, 831, 609, 85, 110, 542, 524, 895,
276, 716, 737, 702, 349, 768, 845, 477, 491, 638, 195, 170, 220, 60, 24, 766, 552, 409,
450, 381, 698, 513, 666, 955, 982, 252, 390, 340, 441, 120, 49, 509, 81, 818, 900, 762,
372, 3, 308, 886, 941, 505, 781, 680, 882, 241, 99, 1019, 163, 613, 777, 500, 744, 7, 616,
748, 859, 1010, 538, 336, 741, 482, 199, 1014, 326, 202, 531, 1001, 464, 14, 209, 473,
695, 996, 53, 673, 459, 964, 399, 1005, 652, 404, 39, 978, 928, 28, 418, 946, 367, 969,
106, 322, 918, 905, 799, 987, 281, 809, 78, 932, 832, 56, 836, 868, 734, 914, 213, 645,
813, 786, 574, 950, 563, 595, 156, 841, 641, 113, 648, 712, 445, 804, 427, 267, 602, 549,
124, 877, 103, 167, 313, 659, 258, 227, 272, 400, 891, 584, 854, 535, 181, 74, 248, 730,
206, 335, 627, 294, 517, 454, 545, 800, 759, 145, 684, 46, 363, 149, 496, 436, 413, 670,
231, 588, 10, 909, 67, 577, 495, 290, 345, 92, 727, 299, 992, 873, 827, 317, 463, 152, 21,
795, 135, 131, 991, 581, 691, 184, 431, 599, 960, 723, 631, 634, 927, 304, 42, 567, 271,
262, 959, 138, 358, 368, 863, 174, 896, 422, 238, 245, 831, 609, 85, 110, 542, 524, 895,
276, 716, 737, 702, 349, 768, 845, 477, 491, 638, 195, 170, 220, 60, 24, 766, 552, 409,
450, 381, 698, 513, 666, 955, 982, 252, 390, 340, 441, 120, 49, 509, 81, 818, 900, 762,
372, 3, 308, 886, 941, 505, 781, 680, 882, 241, 99, 1019, 163, 613, 777, 500, 744, 7, 616,
748, 859, 1010, 538, 336, 741, 482, 199, 1014, 326, 202, 531, 1001, 464, 14, 209, 473,
695, 996, 53, 673, 459, 964, 399, 1005, 652, 404, 39, 978, 928, 28, 418, 946, 367, 969,
106, 322, 918, 905, 799, 987, 281, 809, 78, 932, 832, 56, 836, 868, 734, 914, 213, 645,
813, 786, 574, 950, 563, 595, 156, 841, 641, 113, 648, 712, 445, 804, 427, 267, 602, 549,
124, 877, 103, 167, 313, 659, 258, 227, 272, 400, 891, 584, 854, 535, 181, 74, 248, 730,
206, 335, 627, 294, 517, 454, 545, 800, 759, 145, 684, 46, 363, 149, 496, 436, 413, 670,
231, 588, 10, 909, 67, 577, 495, 290, 345, 92, 727, 299, 992, 873, 827, 317, 463, 152, 21,
795, 135, 131, 991, 581, 691, 184, 431, 599, 960, 723, 631, 634, 927, 304, 42, 567, 271,
262, 959, 138, 358, 368, 863, 174, 896, 422, 238, 245, 831, 609, 85, 110, 542, 524, 895,
276, 716, 737, 702, 349, 768, 845, 477, 491, 638, 195, 170, 220, 60, 24, 766, 552, 409,
450, 381, 698, 513, 666, 955, 982, 252, 390, 340, 441, 120, 49, 509, 81, 818, 900, 762,
372, 3, 308, 886, 941, 505, 781, 680, 882, 241, 99, 1019, 163, 613, 777, 500, 744, 7, 616,
748, 859, 1010, 538, 336, 741, 482, 199, 1014, 326, 202, 531, 1001, 464, 14, 209, 473,
695, 996, 53, 673, 459, 964, 399, 1005, 652, 404, 39, 978, 928, 28, 418, 946, 367, 969,
106, 322, 918, 905, 799, 987, 281, 809, 78, 932, 832, 56, 836, 868, 734, 914, 213, 645,
813, 786, 574, 950, 563, 595, 156, 841, 641, 113, 648, 712, 445, 804, 427, 267, 602, 549,
124, 877, 103, 167, 313, 659, 258, 227, 272, 400, 891, 584, 854, 535, 181, 74, 248, 730,
206, 335, 627, 294, 517, 454, 545, 800, 759, 145, 684, 46, 363, 149, 496, 436, 413, 670,
231, 588, 10, 909, 67, 577, 495, 290, 345, 92, 727, 299, 992, 873, 827, 317, 463, 152, 21,
795, 135, 131, 991, 581, 691, 184, 431, 599, 960, 723, 631, 634, 927, 304, 42, 567, 271,
262, 959, 138, 358, 368, 863, 174, 896, 422, 238, 245, 831, 609, 85, 110, 542, 524, 895,
276, 716, 737, 702, 349, 768, 845, 477, 491, 638, 195, 170, 220, 60, 24, 766, 552, 409
```

Алгоритм 5. Нелинейная комбинация РСЛОС.

Описание алгоритма

Генератор Геффа является примером нелинейной комбинации РСЛОС. В этом генераторе используются три РСЛОС R_1 , R_2 , R_3 , объединённые нелинейным образом. Длины этих регистров L_1 , L_2 , L_3 — попарно простые числа.

Нелинейная функция генератора: $f(x_1, x_2, x_3) = R_1(x_1)R_2(x_2) \oplus (1 + R_2(x_2))R_3(x_3) = R_1(x_1)R_2(x_2) \oplus R_2(x_2)R_3(x_3) \oplus R_3(x_3)$.

Параметры запуска программы

`prng.exe /g:nfsr /i:R1,R2,R3,w,x1,x2,x3`

Исходный текст программы

```
"""
Реализация нелинейной комбинации РСЛОС метода генерации
псевдослучайных чисел
"""

def excep(R : str):
    for s in R:
        if s != '0':
            if s != '1':
                raise Exception

def nfsr_rand_method(m : int,
                    R1 : str,
                    R2 : str,
                    R3 : str,
                    w : int,
                    x1 : int,
                    x2 : int,
                    x3 : int,
                    size : int = 10000):
    r = [0 for _ in range(size)]
    excep(R1)
    excep(R2)
    excep(R3)
    R1_coeff = [i for i in range(len(R1) - 1, -1, -1) if R1[i] == '1']
    R2_coeff = [i for i in range(len(R2) - 1, -1, -1) if R2[i] == '1']
```

```

R3_coeff = [i for i in range(len(R3) - 1, -1, -1) if R3[i] == '1']
for i in range(size):
    xor1, xor2, xor3 = 0, 0, 0
    current = 0
    for _ in range(w):
        n = max(len(R1_coeff), len(R2_coeff), len(R3_coeff))
        for j in range(n):
            if j < len(R1_coeff):
                xor1 ^= x1 >> R1_coeff[j]
            if j < len(R2_coeff):
                xor2 ^= x2 >> R2_coeff[j]
            if j < len(R3_coeff):
                xor3 ^= x3 >> R3_coeff[j]
        x1 >>= 1
        x2 >>= 1
        x3 >>= 1
        x1 |= (xor1 & 1) << (R1_coeff[0] - 1)
        x2 |= (xor2 & 1) << (R2_coeff[0] - 1)
        x3 |= (xor3 & 1) << (R3_coeff[0] - 1)
        output_bit = ((x1 & x2) ^ (x2 & x3) ^ x3) & 1
        current <<= 1
        current ^= output_bit
    r[i] = current % m
return r

```

Пример работы программы

Введём следующие параметры для запуска программы: prng.exe /g:nfsr /i:1000001010011,1000000000000011,100011101,1024,7497,49311,345 /n:10000 /f:rnd_nfsr.dat /m:1024.

rnd_nfsr.dat	
1	496, 761, 514, 826, 515, 908, 716, 676, 527, 449, 122, 771, 696, 9, 952, 424, 942, 647, 773, 763, 22, 630, 754, 216, 686, 802, 623, 448, 254, 286, 958, 561, 1016, 644, 931, 541, 644, 186, 16, 860, 449, 891, 898, 493, 13, 420, 498, 210, 770, 587, 840, 41, 418, 175, 389, 411, 530, 826, 654, 491, 556, 954, 674, 964, 233, 306, 800, 905, 74, 740, 270, 201, 508, 811, 94, 632, 673, 353, 676, 932, 782, 193, 248, 50, 176, 131, 972, 746, 749, 143, 459, 126, 914, 569, 788, 704, 702, 925, 654, 23, 249, 342, 336, 63, 834, 505, 108, 932, 352, 723, 130, 617, 846, 521, 934, 423, 141, 985, 570, 258, 878, 875, 888, 702, 176, 716, 192, 698, 528, 652, 67, 836, 558, 964, 30, 323, 254, 320, 824, 33, 898, 140, 941, 719, 277, 499, 146, 696, 654, 616, 702, 654, 175, 769, 255, 19, 760, 645, 1008, 821, 653, 134, 352, 223, 82, 572, 835, 968, 556, 933, 6, 465, 506, 67, 808, 547, 450, 686, 397, 927, 723, 427, 334, 490, 573, 762, 160, 1004, 704, 928, 545, 399, 96, 592, 590, 196, 302, 523, 476, 362, 274, 881, 643, 450, 780, 573, 924, 487, 251, 24, 652, 579, 846, 654, 428, 543, 461, 179, 208, 537, 530, 848, 652, 174, 684, 455, 195, 22, 632, 585, 840, 676, 804, 659, 384, 235, 256, 568, 3, 1002, 396, 957, 651, 419, 459, 418, 572, 762, 168, 748, 672, 931, 769, 137, 18, 624, 527, 984, 510, 783, 173, 328, 1019, 83, 546, 967, 904, 557, 813, 71, 464, 250, 74, 554, 563, 202, 174, 957, 958, 161, 765, 68, 1008, 771, 972, 518, 868, 30, 448, 63, 322, 825, 8, 992, 56, 729, 394, 195, 461, 40, 306, 167, 396, 651, 938, 682, 461, 511, 30, 762, 674, 968, 676, 936, 652, 449, 219, 592, 524, 323, 968, 554, 765, 270, 465, 481, 658, 42, 774, 716, 44, 959, 242, 153, 255, 98, 634, 131, 462, 556, 957, 655, 476, 225, 434, 825, 520, 334, 748, 428, 715, 141, 123, 24, 520, 899, 840, 699, 910, 655, 961, 235, 274, 568, 516, 904, 548, 813, 654, 449, 251, 18, 560, 673, 712, 942, 905, 653, 193, 767, 30, 720, 371, 961, 700, 841, 702, 449, 210, 198, 778, 535, 408, 806, 507, 156, 708, 739, 126, 156, 10, 876, 684, 421, 670, 197, 498, 272, 520, 2, 872, 158, 456, 159, 451, 251, 88, 672, 256, 961, 170, 717, 654, 1, 499, 147, 536, 659, 672, 552, 1007, 1007, 529, 251, 176, 760, 642, 986, 896, 941, 719, 756, 43, 400, 382, 771, 492, 200, 944, 331, 128, 230, 282, 537, 280, 869, 250, 652, 654, 963, 127, 150, 570, 548, 936, 165, 957, 711, 464, 505, 10, 50, 561, 478, 460, 431, 715, 505, 127, 20, 624, 707, 448, 555, 844, 15, 466, 247, 136, 922, 562, 906, 910, 495, 700, 737, 177, 236, 728, 514, 832, 908, 356, 540, 204, 242, 274, 777, 88, 840, 696, 485, 651, 455, 142, 598, 56, 519, 992, 172, 940, 642, 88, 235, 50, 824, 675, 504, 1020, 749, 971, 405, 649, 18, 560, 590, 456, 173, 928, 987, 77, 184, 48, 657, 131, 585, 748, 685, 943, 451, 201, 914, 568, 514, 974, 236, 557, 699, 128, 251, 528, 560, 515, 904, 672, 892, 655, 448, 251, 259, 120, 768, 1000, 228, 653, 399, 398, 93, 50, 305, 659, 648, 716, 686, 686, 463, 187, 822, 826, 545, 1020, 708, 952, 917, 140, 31, 624, 286, 195, 271, 584, 505, 590, 453, 368, 147, 154, 523, 844, 680, 936, 655, 457, 987, 530, 554, 326, 489, 700, 761, 131, 449, 192, 946, 544, 901, 960, 636, 653, 130, 347, 331, 82, 552, 929, 480, 686, 909, 654, 321, 255, 83, 688, 643, 968, 558, 685, 142, 467, 510, 786, 568, 565, 704, 956, 941, 652, 7, 763, 286, 720, 15, 840, 441, 620, 901, 368, 723, 146, 538, 582, 136, 936, 935, 909, 985, 676, 58, 222, 99, 988, 760, 673, 717, 448, 1010, 514, 957
Normal t length: 49 194 lines: 1 Ln: 1 Col: 1 Pos: 1 Windows (CR LF) UTF-8 INS	

Алгоритм 6. Вихрь Мерсенна.

Описание алгоритма

По сути данный ГПСЧ является РСЛОС, состоящим из 624 ячеек по 32 бита. Метод Вихрь Мерсенна позволяет генерировать последовательность двоичных псевдослучайных целых w -битовых чисел в соответствии со следующей рекуррентной формулой

$$X_{n+p} = X_{(n+q)} \oplus (X_n^r | X_{n+1}^l)A \quad (n = 0, 1, 2, \dots)$$

где p, q, r — целые константы, p — степень рекуррентности, $1 \leq q \leq p$;

X_n — w -битовое двоичное целое число;

$(X_n^r | X_{n+1}^l)$ — двоичное целое число, полученное конкатенацией чисел X_n^r и X_{n+1}^l , когда первые $(w - r)$ битов взяты из X_n , а последние r битов из X_{n+1} в том же порядке;

A — матрица размера $w \times w$, состоящая из нулей и единиц, определенная посредством a ;

XA — произведение, при вычислении которого сначала выполняют операцию $X \gg 1$ (сдвига битов на одну позицию вправо), если последний бит X равен 0, а затем, когда последний бит $X = 1$, вычисляют $XA = (X \gg 1) \oplus a$,

$$\begin{aligned} a &= (a_{w-1}, a_{w-2}, \dots, a_0), \\ X &= (x_{w-1}, x_{w-2}, \dots, x_0), \\ A &= \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ \vdots & \dots & \dots & \ddots & \vdots \\ 0 & 0 & & & 1 \\ a_{w-1} & a_{w-2} & \dots & \dots & a_0 \end{pmatrix}. \end{aligned}$$

Алгоритм Вихрь Мерсенна состоит из попеременного выполнения процедур рекурсивной генерации и «закалки». Рекурсивная генерация представляет из себя РСЛОС с дополнительной рекурсивной функцией для потока выходных битов. Операция «закалки» является процедурой, усиливающей равномерность распределения на больших размерностях битовых векторов.

Шаги алгоритма.

Шаг 1а. Инициализируются значения u , h , a по формуле:

$u := (1, 0, \dots, 0)$ — всего $w - r$ бит, $h := (0, 1, \dots, 1)$ — всего r бит, $a := (a_{w-1}, a_{w-2}, \dots, a_0)$ — последняя строка матрицы A .

Шаг 1б. X_0, X_1, \dots, X_{p-1} заполняются начальными значениями.

Шаг 2. Вычисляется $Y := (y_0, y_1, \dots, y_{w-1}) := (X_n^r | X_{n+1}^l)$.

Шаг 3. Вычисляется новое значение X_i :

$X_n := X_{n+q} \bmod p \oplus (Y \gg 1) \oplus a$, если младший бит $y_0 = 1$;

$X_n := X_{n+q} \bmod p \oplus (Y \gg 1) \oplus 0$, если младший бит $y_0 = 0$;

Шаг 4. Вычисляется $X_i T$.

$Y := X_n$,

$Y := Y \oplus (Y \gg u)$,

$Y := Y \oplus ((Y \ll s) \cdot b)$,

$Y := Y \oplus ((Y \ll t) \cdot c)$,

$Z := Y \oplus (Y \gg l)$.

Z подается на выход, как результат.

Шаг 5. $n := (n + 1) \bmod p$. Переход на шаг 2.

Параметры алгоритма были тщательно подобраны создателями с целью достижения наилучших свойств. Параметры p и r выбраны так, что образующий многочлен – примитивный степени 19937. w выбирается по размеру стандартного машинного слова – 32 или 64 бита. Однако для 64-битной версии формула выглядит несколько иначе. Значение последней строки матрицы A выбирается случайным образом и подается на вход алгоритма. Параметры «заковки» подобраны так, чтобы дать в итоге хорошее равномерное распределение.

Параметры алгоритма Вихрь Мерсенна: $p = 624$, $w = 32$, $r = 31$, $q = 397$, $a = 2567483615$ (9908B0DF16), $u = 11$, $s = 7$, $t = 15$, $l = 18$, $b = 2636928640$ (9D2C568016), $c = 4022730752$ (EFC6000016).

Параметры запуска программы

prng.exe /g:mt /i:modulus, X_0

Исходный текст программы

```
"""
Реализация вихря Мерсена метода генерации псевдослучайных чисел
"""

p, w, r, q = 624, 32, 31, 397
u = 1 << (w - r)
h = (1 << (r + 1)) - 1
a, b, c = 0x9908B0DF, 0x9D2C5680, 0xEFC60000
s = 7
t = 15
l = 18

def mt_rand_method(m : int, modulus : int, seed : int, size = 10000):
    x = [0] * p
    x[0] = seed
    for i in range(1, p):
        x[i] = (69069 * x[i - 1]) & ((1 << w + 1) - 1)

    mta = [0, a]
    i = 0
    r = [0 for _ in range(size)]
    for j in range(size):
        y = (x[i] & u) | (x[(i + 1) % p] & h)

        x[i] = x[(i + q) % p] ^ (y >> 1) ^ mta[y & 1]

        y = x[i]
        y ^= y >> u
        y ^= (y << s) & b
        y ^= (y << t) & c

        i = (i + 1) % p

    r[j] = (y ^ (y >> 1)) % modulus
    return [i%m for i in r]
```

Пример работы программы

Введём следующие параметры для запуска программы: prng.exe /g:mt /i:4096,25 /n:10000 /m:1024 /f:rnd_mt.dat.


```
md_mt.dat
1 342, 10, 980, 69, 42, 212, 365, 328, 135, 676, 223, 268, 294, 292, 730, 800, 638, 898,
478, 679, 387, 368, 80, 687, 643, 107, 208, 882, 132, 876, 193, 392, 484, 952, 804, 334,
161, 438, 906, 779, 793, 889, 1013, 526, 528, 626, 248, 564, 5, 941, 78, 734, 854, 465,
1006, 633, 608, 19, 461, 527, 751, 247, 1001, 598, 569, 695, 53, 1019, 191, 856, 320, 725,
204, 767, 635, 174, 1022, 832, 331, 621, 502, 831, 382, 653, 879, 548, 407, 879, 206, 408,
991, 211, 858, 656, 785, 377, 464, 917, 222, 637, 895, 940, 737, 139, 162, 352, 903, 153,
741, 164, 656, 143, 465, 425, 1014, 23, 133, 7, 571, 1, 844, 52, 747, 1019, 642, 345, 533,
56, 466, 430, 174, 331, 56, 980, 537, 853, 364, 399, 384, 576, 233, 897, 313, 539, 2, 809,
248, 484, 47, 430, 469, 653, 622, 286, 766, 94, 862, 328, 688, 631, 412, 285, 184, 216,
347, 112, 234, 82, 40, 582, 936, 77, 219, 407, 936, 425, 399, 530, 541, 268, 183, 752,
802, 376, 272, 744, 978, 986, 1014, 381, 425, 932, 968, 920, 846, 865, 841, 491, 132, 811,
963, 180, 394, 315, 893, 366, 900, 914, 654, 97, 694, 560, 248, 587, 88, 168, 728, 414,
159, 778, 289, 193, 180, 143, 780, 883, 285, 776, 301, 470, 1019, 599, 40, 11, 710, 844,
919, 1013, 290, 220, 726, 219, 494, 192, 939, 824, 429, 674, 440, 168, 736, 498, 840, 832,
188, 873, 729, 988, 158, 734, 16, 555, 10, 472, 993, 296, 669, 938, 614, 263, 35, 746,
578, 466, 402, 271, 764, 574, 314, 501, 547, 1004, 160, 746, 390, 217, 249, 246, 816, 173,
450, 634, 596, 854, 434, 614, 1004, 931, 881, 596, 706, 19, 754, 554, 257, 68, 503, 984,
913, 125, 853, 773, 694, 670, 251, 620, 930, 793, 747, 602, 794, 306, 911, 793, 686, 261,
994, 895, 680, 222, 838, 294, 482, 372, 554, 576, 971, 1016, 453, 185, 112, 95, 152, 948,
328, 223, 760, 341, 120, 468, 151, 769, 238, 781, 157, 984, 103, 356, 143, 975, 769, 1022,
263, 1016, 646, 734, 420, 749, 196, 347, 458, 682, 779, 439, 148, 300, 513, 616, 698, 243,
415, 974, 632, 252, 375, 149, 11, 852, 454, 421, 515, 747, 474, 234, 16, 190, 405, 133,
314, 398, 756, 88, 223, 341, 206, 801, 295, 120, 416, 190, 523, 351, 409, 853, 690, 985,
379, 53, 19, 607, 838, 131, 547, 133, 758, 100, 897, 666, 556, 239, 840, 839, 660, 690,
522, 636, 824, 118, 322, 327, 114, 36, 11, 824, 633, 231, 955, 62, 247, 343, 240, 830,
561, 786, 804, 53, 343, 576, 921, 922, 402, 597, 140, 118, 734, 745, 963, 524, 315, 628,
170, 684, 897, 732, 195, 1014, 882, 698, 635, 68, 490, 448, 77, 584, 883, 593, 694, 121,
462, 910, 97, 382, 644, 507, 564, 1018, 58, 342, 366, 818, 553, 454, 360, 455, 845, 992,
34, 456, 478, 1005, 368, 617, 898, 818, 258, 14, 780, 190, 764, 723, 316, 103, 10, 530,
515, 378, 3, 3, 69, 184, 661, 152, 620, 24, 52, 239, 571, 902, 77, 0, 919, 275, 616, 737,
706, 222, 538, 309, 918, 891, 20, 509, 992, 737, 426, 922, 148, 494, 127, 300, 272, 517,
161, 643, 336, 391, 222, 313, 416, 623, 1015, 738, 216, 28, 296, 688, 346, 76, 765, 832,
723, 745, 904, 668, 288, 444, 880, 485, 551, 873, 359, 672, 916, 705, 555, 301, 574, 183,
866, 778, 864, 641, 66, 43, 745, 230, 56, 290, 279, 114, 508, 790, 978, 263, 861, 172, 19,
382, 559, 753, 696, 996, 975, 72, 167, 75, 959, 722, 840, 98, 721, 1005, 668, 732, 36,
966, 955, 961, 239, 710, 988, 262, 242, 899, 850, 601, 547, 352, 756, 255, 768, 289, 608,
969, 978, 560, 625, 192, 122, 688, 888, 649, 954, 947, 612, 555, 307, 526, 527, 185, 303,
884 798 398 684 14 203 34 608 39 242 825 811 1018 264 174 286 525 195
```

Normal t length : 49 156 lines : 1 Ln : 1 Col : 2 109 Pos : 2 109 Windows (CR LF) UTF-8 INS

Алгоритм 7. RC4.

Описание алгоритма

1. Инициализация $S_i, i = 0, 1, \dots, 255$.
 - a) for $i = 0$ to 255: $S_i = i$;
 - b) $j = 0$;
 - c) for $i = 0$ to 255: $j = (j + S_i + K_i) \bmod 256$; Swap(S_i, S_j)
2. $i = 0, j = 0$.
3. Итерация алгоритма:
 - a) $i = (i + 1) \bmod 256$;
 - b) $j = (j + S_i) \bmod 256$;
 - c) Swap(S_i, S_j);
 - d) $t = (S_i + S_j) \bmod 256$;
 - e) $K = S_t$;

Параметры запуска программы

prng /g:rc4 /i: $K_0, K_1, K_2, \dots, K_{255}$

Исходный текст программы

```
"""
Реализация линейно-конгруэнтного метода генерации псевдослучайных чисел
"""

def rc4_rand_method(m : int, K : list[int], size=10000):
    if len(K) != 256:
        raise Exception
    s = [i for i in range(256)]
    j = 0
    for i in range(256):
        j = (j + s[i] + K[i]) % 256
        s[i], s[j] = s[j], s[i]

    i = j = 0
    r = [0 for _ in range(size)]
    for d in range(size):
        k, l = 0, 0
        for _ in range(2):
            k = 1
```



```

        i = (i + 1) % 256
        j = (j + s[i]) % 256
        s[i], s[j] = s[j], s[i]
        t = (s[i] + s[j]) % 256
        l = s[t]
        r[d] = int(str(l) + str(k)) % m
    return r

```

Пример работы программы

Введём следующие параметры для запуска программы: `prng.exe /g:rc4 /i:3179,3298,3097,2987,2258,2437,195,583,1623,2324,1886,3533,1935,254,1697,2568,2181,2266,3523,3830,535,3541,1025,2103,290,3932,3481,909,4047,2991,2586,129,2338,1775,706,1232,2149,3662,2979,175,3241,2746,431,1137,2249,806,3589,22,2536,3305,3375,844,4047,3642,3293,3153,2464,437,488,3122,2235,771,828,3208,2879,3482,2070,1233,11,1985,2594,1762,3934,642,2227,3319,1403,2038,4071,3533,1740,3480,2488,2788,2784,890,264,163,4012,662,762,1410,2948,1937,2593,369,2871,1094,1347,4041,1516,3187,2897,3381,342,1125,1350,512,3698,1854,692,2027,1682,3734,4003,753,1349,20,2203,3273,3287,3889,458,691,2666,2126,617,2832,142,2853,3938,843,2352,2377,435,1051,1383,4070,3333,1726,1382,3098,2874,2303,2592,2034,874,3591,2833,1628,3144,3702,2086,3820,783,966,4039,1654,3876,3345,3243,3645,49,1549,22,1945,3010,2182,3350,3038,3687,1545,3194,1693,3793,3947,3254,3438,1337,2986,1500,1970,1130,2007,2477,3272,2090,746,2576,78,1204,3821,2162,1001,3963,2814,273,2751,419,735,175,1385,1844,63,264,1379,2546,3196,863,1185,3727,3353,662,1172,2545,2122,1470,2940,1036,3108,3717,1266,2252,2540,2065,993,2492,1469,187,2773,4091,2432,2003,509,1930,2326,2440,2882,2673,562,1563,2950,367,2366,2763,3293,2654,2122,1836,4089,1000,3463,3052,9,1078,2453 /n:10000 /m:1024 /f:rnd_rc4.dat.`

md_rc4.dat	
1	845, 820, 503, 829, 446, 121, 905, 337, 674, 608, 318, 441, 325, 637, 710, 820, 183, 806, 686, 538, 175, 94, 559, 815, 206, 111, 39, 831, 137, 966, 49, 641, 790, 912, 295, 758, 242, 596, 1012, 385, 919, 30, 850, 1012, 729, 130, 145, 251, 890, 888, 129, 672, 585, 317, 104, 322, 447, 685, 365, 137, 605, 183, 407, 6, 509, 232, 782, 772, 567, 455, 409, 286, 570, 778, 381, 994, 908, 119, 280, 870, 510, 115, 590, 449, 444, 542, 812, 732, 368, 194, 511, 805, 95, 974, 256, 649, 619, 542, 982, 737, 210, 12, 159, 915, 153, 1020, 986, 38, 842, 339, 424, 905, 4, 506, 463, 849, 286, 389, 420, 835, 643, 882, 623, 340, 624, 984, 668, 766, 26, 463, 305, 886, 729, 362, 998, 294, 204, 743, 792, 306, 202, 420, 974, 259, 283, 981, 241, 35, 322, 744, 331, 932, 138, 622, 162, 319, 518, 831, 960, 265, 373, 979, 934, 307, 602, 392, 372, 510, 628, 722, 709, 66, 823, 617, 393, 869, 597, 401, 508, 445, 942, 351, 349, 945, 161, 78, 807, 498, 748, 430, 562, 198, 723, 292, 223, 560, 348, 873, 1004, 803, 8, 482, 184, 313, 77, 157, 913, 309, 1007, 455, 574, 964, 146, 684, 592, 243, 171, 309, 670, 257, 174, 982, 574, 772, 173, 924, 932, 151, 726, 775, 64, 401, 19, 574, 49, 621, 413, 207, 996, 263, 65, 417, 244, 931, 876, 828, 146, 749, 753, 283, 648, 276, 374, 14, 385, 744, 518, 517, 662, 972, 339, 381, 778, 912, 336, 551, 390, 969, 246, 825, 517, 829, 290, 269, 98, 916, 476, 997, 735, 822, 576, 415, 944, 347, 180, 165, 684, 176, 100, 841, 569, 45, 745, 162, 465, 63, 310, 979, 950, 919, 611, 793, 368, 222, 220, 985, 327, 726, 727, 199, 596, 124, 494, 306, 919, 1001, 159, 400, 329, 453, 993, 1013, 332, 615, 195, 858, 358, 346, 670, 131, 494, 945, 10, 174, 459, 999, 559, 263, 413, 988, 995, 3, 422, 365, 581, 85, 933, 396, 93, 83, 1005, 902, 641, 318, 50, 150, 988, 78, 223, 981, 632, 1000, 269, 650, 519, 914, 550, 116, 584, 642, 278, 930, 769, 654, 595, 904, 654, 374, 691, 815, 985, 78, 900, 607, 230, 569, 974, 816, 853, 458, 551, 643, 933, 289, 766, 920, 664, 188, 388, 790, 185, 137, 245, 21, 604, 638, 934, 413, 439, 428, 764, 923, 1006, 943, 579, 174, 951, 343, 538, 921, 170, 551, 162, 668, 314, 213, 677, 771, 152, 174, 961, 141, 969, 426, 745, 873, 579, 612, 897, 565, 933, 1019, 0, 536, 333, 480, 755, 966, 583, 816, 453, 492, 28, 882, 808, 589, 982, 321, 366, 126, 62, 287, 109, 191, 269, 152, 355, 905, 748, 607, 870, 502, 794, 394, 749, 197, 645, 517, 512, 713, 78, 90, 601, 397, 186, 102, 263, 110, 790, 479, 853, 549, 30, 814, 426, 194, 40, 623, 523, 241, 27, 561, 928, 71, 436, 292, 979, 1010, 307, 724, 717, 592, 157, 911, 790, 918, 234, 68, 226, 820, 191, 713, 943, 962, 857, 277, 704, 891, 442, 533, 261, 813, 788, 586, 630, 862, 223, 148, 780, 557, 136, 140, 433, 668, 871, 419, 686, 405, 387, 661, 119, 887, 912, 102, 13, 150, 846, 508, 169, 196, 218, 454, 726, 146, 177, 252, 726, 900, 528, 1013, 867, 881, 691, 763, 211, 173, 181, 256, 264, 618, 307, 585, 81, 39, 912, 449, 821, 520, 942, 1021, 732, 599, 154, 972, 38, 393, 509, 982, 330, 587, 365, 18, 819, 846, 523, 388, 830, 437, 47, 596, 380, 23, 689, 227, 62, 408, 94, 255, 209, 917, 976, 363, 319, 816, 460, 355, 166, 708, 111, 627, 15, 939, 731, 939, 924, 199, 254, 476, 672, 179, 374, 641, 437, 664, 682, 255, 482, 354, 763, 621, 594, 129, 127, 411, 522, 768, 994, 431, 484, 923, 630, 704, 1014, 536, 775, 696, 421, 347, 327, 534, 159, 710, 679, 205, 191, 963, 562, 721, 103, 778, 46, 819, 274, 1005, 1021
Normal t length : 49 156 lines : 1 Ln : 1 Col : 1 Pos : 1 Windows (CR LF) UTF-8 INS	

Алгоритм 8. ГПСЧ на основе RSA.

Описание алгоритма

1. Сгенерировать два секретных простых числа p и q , а также $n = pq$ и $f = (p - 1)(q - 1)$. Выбрать случайное целое число e , $1 < e < f$, такое что $\text{НОД}(e, f) = 1$.
2. Выбрать случайное целое x_0 – начальный вектор из интервала $[1, n - 1]$.
3. For $i = 1$ to l do
 - a. $x_i \leftarrow x_{i-1}^e \bmod n$;
 - b. $z_i \leftarrow$ последний значащий бит x_i .
4. Вернуть z_1, z_2, \dots, z_l .

Параметры запуска программы

prng.exe /g:rsa /i:n,e,w,x

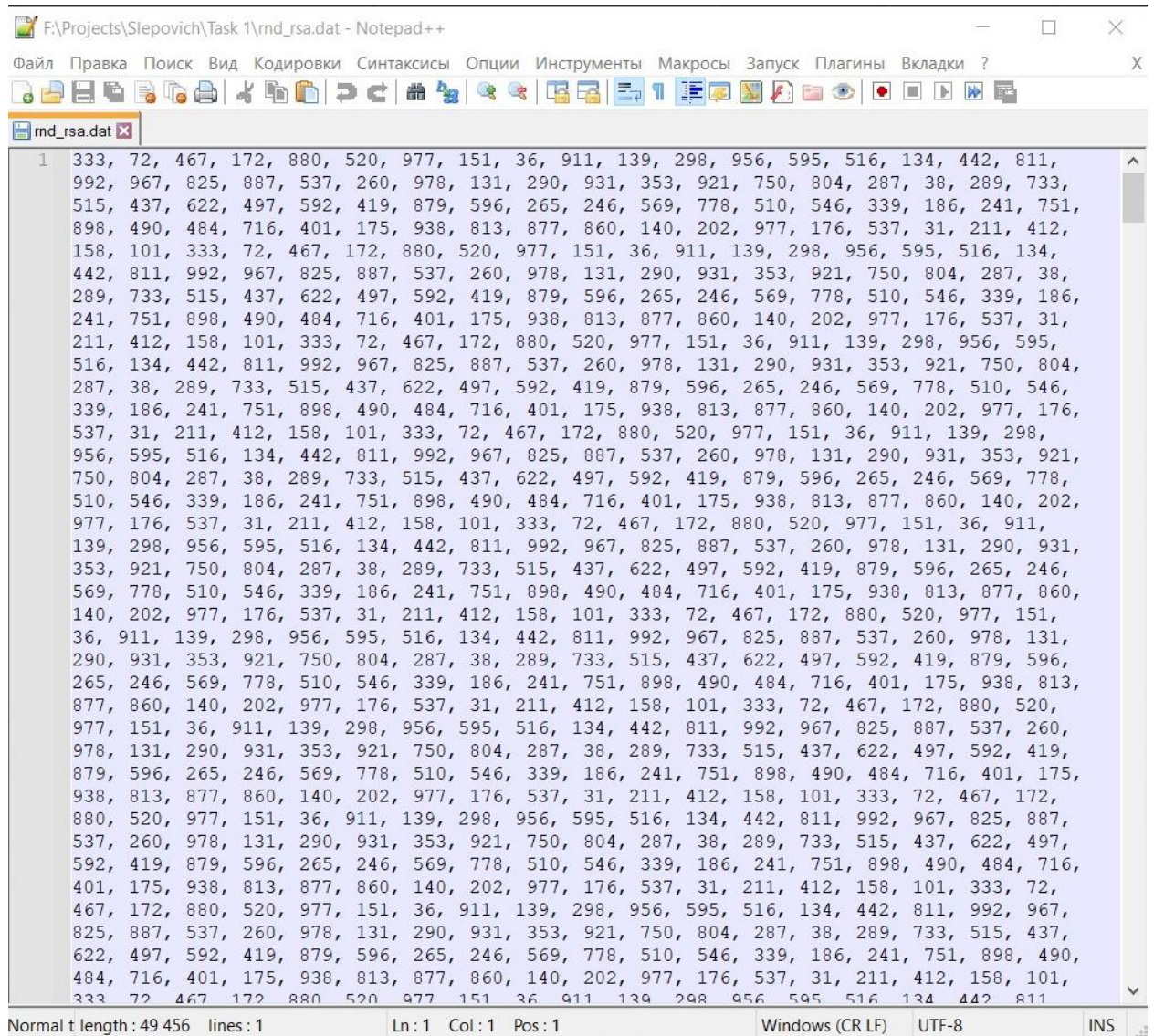
Исходный текст программы

```
"""
Реализация RSA метода генерации псевдослучайных чисел
"""

def RSA_rand_method(
    m: int,
    n: int,
    e: int,
    w: int,
    x: int,
    size: int = 10000
):
    r = []
    for _ in range(size):
        current = 0
        for _ in range(w):
            x = pow(x, e, n)
            current <<= 1
            current ^= x & 1
        r.append(current % m)
    return r
```

Пример работы программы

Введём следующие параметры для запуска программы: prng.exe /g:rsa
/i:12709189,53,300,25 /n:10000 /m:1024 /f:rnd_rsa.dat.



```
F:\Projects\Slepovich\Task 1\rnd_rsa.dat - Notepad++
Файл Правка Поиск Вид Кодировки Синтаксисы Опции Инструменты Макросы Запуск Плагины Вкладки ?
rnd_rsa.dat
1 333, 72, 467, 172, 880, 520, 977, 151, 36, 911, 139, 298, 956, 595, 516, 134, 442, 811,
992, 967, 825, 887, 537, 260, 978, 131, 290, 931, 353, 921, 750, 804, 287, 38, 289, 733,
515, 437, 622, 497, 592, 419, 879, 596, 265, 246, 569, 778, 510, 546, 339, 186, 241, 751,
898, 490, 484, 716, 401, 175, 938, 813, 877, 860, 140, 202, 977, 176, 537, 31, 211, 412,
158, 101, 333, 72, 467, 172, 880, 520, 977, 151, 36, 911, 139, 298, 956, 595, 516, 134,
442, 811, 992, 967, 825, 887, 537, 260, 978, 131, 290, 931, 353, 921, 750, 804, 287, 38,
289, 733, 515, 437, 622, 497, 592, 419, 879, 596, 265, 246, 569, 778, 510, 546, 339, 186,
241, 751, 898, 490, 484, 716, 401, 175, 938, 813, 877, 860, 140, 202, 977, 176, 537, 31,
211, 412, 158, 101, 333, 72, 467, 172, 880, 520, 977, 151, 36, 911, 139, 298, 956, 595,
516, 134, 442, 811, 992, 967, 825, 887, 537, 260, 978, 131, 290, 931, 353, 921, 750, 804,
287, 38, 289, 733, 515, 437, 622, 497, 592, 419, 879, 596, 265, 246, 569, 778, 510, 546,
339, 186, 241, 751, 898, 490, 484, 716, 401, 175, 938, 813, 877, 860, 140, 202, 977, 176,
537, 31, 211, 412, 158, 101, 333, 72, 467, 172, 880, 520, 977, 151, 36, 911, 139, 298,
956, 595, 516, 134, 442, 811, 992, 967, 825, 887, 537, 260, 978, 131, 290, 931, 353, 921,
750, 804, 287, 38, 289, 733, 515, 437, 622, 497, 592, 419, 879, 596, 265, 246, 569, 778,
510, 546, 339, 186, 241, 751, 898, 490, 484, 716, 401, 175, 938, 813, 877, 860, 140, 202,
977, 176, 537, 31, 211, 412, 158, 101, 333, 72, 467, 172, 880, 520, 977, 151, 36, 911,
139, 298, 956, 595, 516, 134, 442, 811, 992, 967, 825, 887, 537, 260, 978, 131, 290, 931,
353, 921, 750, 804, 287, 38, 289, 733, 515, 437, 622, 497, 592, 419, 879, 596, 265, 246,
569, 778, 510, 546, 339, 186, 241, 751, 898, 490, 484, 716, 401, 175, 938, 813, 877, 860,
140, 202, 977, 176, 537, 31, 211, 412, 158, 101, 333, 72, 467, 172, 880, 520, 977, 151,
36, 911, 139, 298, 956, 595, 516, 134, 442, 811, 992, 967, 825, 887, 537, 260, 978, 131,
290, 931, 353, 921, 750, 804, 287, 38, 289, 733, 515, 437, 622, 497, 592, 419, 879, 596,
265, 246, 569, 778, 510, 546, 339, 186, 241, 751, 898, 490, 484, 716, 401, 175, 938, 813,
877, 860, 140, 202, 977, 176, 537, 31, 211, 412, 158, 101, 333, 72, 467, 172, 880, 520,
977, 151, 36, 911, 139, 298, 956, 595, 516, 134, 442, 811, 992, 967, 825, 887, 537, 260,
978, 131, 290, 931, 353, 921, 750, 804, 287, 38, 289, 733, 515, 437, 622, 497, 592, 419,
879, 596, 265, 246, 569, 778, 510, 546, 339, 186, 241, 751, 898, 490, 484, 716, 401, 175,
938, 813, 877, 860, 140, 202, 977, 176, 537, 31, 211, 412, 158, 101, 333, 72, 467, 172,
880, 520, 977, 151, 36, 911, 139, 298, 956, 595, 516, 134, 442, 811, 992, 967, 825, 887,
537, 260, 978, 131, 290, 931, 353, 921, 750, 804, 287, 38, 289, 733, 515, 437, 622, 497,
592, 419, 879, 596, 265, 246, 569, 778, 510, 546, 339, 186, 241, 751, 898, 490, 484, 716,
401, 175, 938, 813, 877, 860, 140, 202, 977, 176, 537, 31, 211, 412, 158, 101, 333, 72,
467, 172, 880, 520, 977, 151, 36, 911, 139, 298, 956, 595, 516, 134, 442, 811, 992, 967,
825, 887, 537, 260, 978, 131, 290, 931, 353, 921, 750, 804, 287, 38, 289, 733, 515, 437,
622, 497, 592, 419, 879, 596, 265, 246, 569, 778, 510, 546, 339, 186, 241, 751, 898, 490,
484, 716, 401, 175, 938, 813, 877, 860, 140, 202, 977, 176, 537, 31, 211, 412, 158, 101,
333, 72, 467, 172, 880, 520, 977, 151, 36, 911, 139, 298, 956, 595, 516, 134, 442, 811,
333 72 467 172 880 520 977 151 36 911 139 298 956 595 516 134 442 811
```

Normal t length : 49 456 lines : 1 Ln: 1 Col: 1 Pos: 1 Windows (CR LF) UTF-8 INS

Алгоритм 9. Алгоритм Блюма-Блюма-Шуба.

Описание алгоритма

1. Сгенерировать два простых числа p и q , сравнимых с 3 по модулю 4.
Это гарантирует, что каждый квадратичный вычет имеет один квадратный корень, который также является квадратичным вычетом.
Произведение этих чисел – $n = pq$ является целым числом Блюма.
Выберем другое случайное целое число x , взаимно простое с n .
2. Вычислим $x_0 = x^2 \bmod n$, которое будет начальным вектором.
3. For $i = 1$ to l do
 - a. $x_i \leftarrow x_{i-1}^2 \bmod n$.
 - b. $z_i \leftarrow$ последний значащий бит x_i
4. Вернуть z_1, z_2, \dots, z_l .

Параметры запуска программы

prng.exe /g:bbs /i:x₀

Исходный текст программы

```
"""
Реализация Блюма-Блюма-Шуба метода генерации псевдослучайных чисел
"""
import math

p, q, n = 127, 131, 16637

def bbs_rand_method(m: int, x: int, size: int = 10000):
    if math.gcd(x, n) != 1 or x < 1 or x >= n:
        raise Exception
    r = []
    for _ in range(size):
        current = 0
        for _ in range(n.bit_length()):
            x = pow(x, 2, n)
            current <<= 1
            current ^= (x & 1)
        r.append(current % m)
    return r
```

Пример работы программы

The image shows a Notepad++ window with the title bar 'F:\Projects\Slepevich\Task 1\md_bbs.dat - Notepad++'. The menu bar includes 'Файл', 'Правка', 'Поиск', 'Вид', 'Кодировки', 'Синтаксисы', 'Опции', 'Инструменты', 'Макросы', 'Запуск', 'Плагин', 'Вкладки', and '?'. The toolbar contains various icons for file operations, editing, and searching. The main text area displays a single line of text, 'md_bbs.dat', which is a long sequence of the numbers 365, 878, 885, and 941. The status bar at the bottom shows 'Normal t length : 49 998 lines : 1'.