

МИНОБРНАУКИ РОССИИ

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
ОТЧЁТ ПО ПРАКТИЧЕСКОМУ КУРСУ

студента 4 курса 431 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий

Яхина Шамиля Илдусовича

Преподаватель

Ассистент

Н.А. Артемова

подпись, дата

Саратов 2023

1 Генератор псевдослучайных чисел

Описание задания:

Создайте программу для генерации псевдослучайных величин следующими алгоритмами:

1. Линейный конгруэнтный метод;
2. Аддитивный метод;
3. Пятипараметрический метод;
4. Регистр сдвига с обратной связью (РСЛОС);
5. Нелинейная комбинация РСЛОС;
6. Вихрь Мерсенна;
7. RC4;
8. ГПСЧ на основе RSA;
9. Алгоритм Блюма-Блюма-Шуба.

1.1 Линейный конгруэнтный метод

Описание алгоритма:

Последовательность ПСЧ, получаемая по формуле:

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

называется *линейной конгруэнтной последовательностью (ЛКП)*.

В его основе лежит выбор четырех ключевых чисел:

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

Инициализирующий вектор: модуль, множитель, приращение, начальное значение.

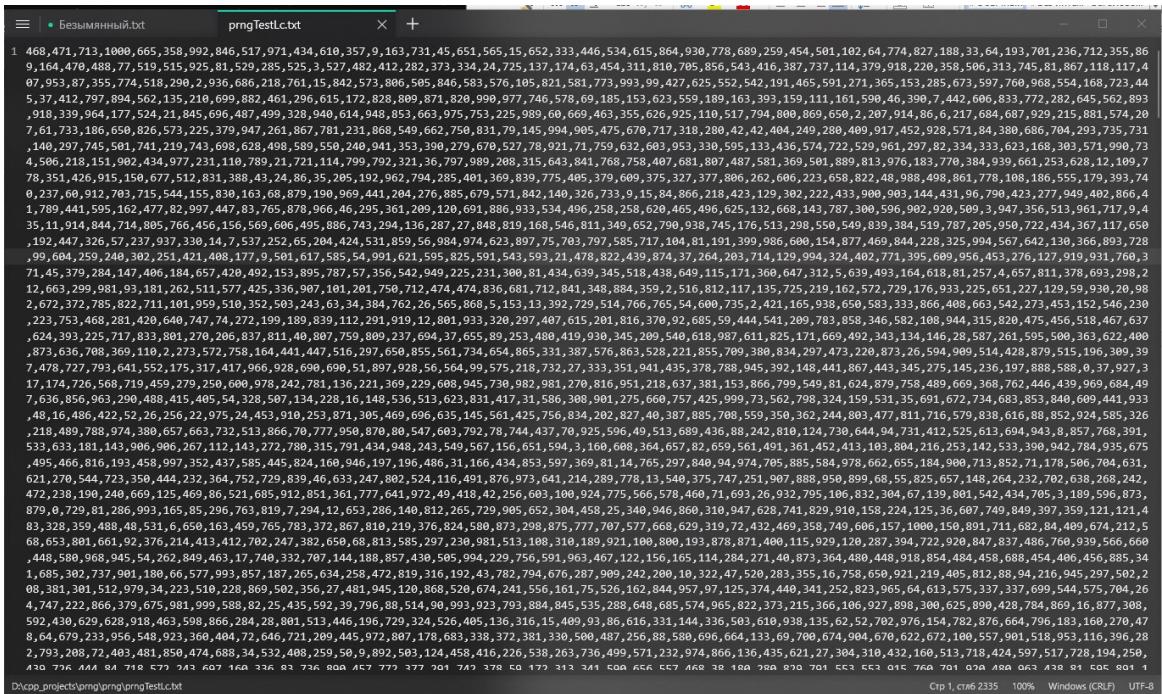
Параметры запуска программы:

`prng.exe /n:16000 /g:lc /i:12960,1741,2731,1 /f:prngTestLc.txt`

```
D:\cpp_projects\prng\prng.exe /n:16000 /g:lc /i:12960,1741,2731,1 /f:prngTestLc.txt
The process of generating pseudo-random numbers:
10% completed
20% completed
30% completed
40% completed
50% completed
60% completed
70% completed
80% completed
90% completed
100% completed

Generation method: lc
Count of generated numbers: 16000
Full name of the output file: prngTestLc.txt
```

Рисунок 1 – Запуск программы



```

        cout << "\nError. The parameters do not meet the requirements.\n";
        return;
    }
    if (nod(c, m) != 1) {
        cout << "\nThe parameters do not meet the requirements of Theorem
        ↪ 3.1 (Item 1).\n";
    }
    if (!(kratP(a - 1, m))) {
        cout << "\nThe parameters do not meet the requirements of Theorem
        ↪ 3.1 (Item 2).\n";
    }
    if (m % 4 == 0 && (a - 1) % 4 != 0) {
        cout << "\nThe parameters do not meet the requirements of Theorem 3.1 (Item
        ↪ 3).\n";
    }
}

vector<int> showsVector = progressVector(NumCount);
int check_progress_index = 0;

ofstream f;
f.open(fileName, ios::out);
for (int i = 0; i < NumCount; i++)
{
    x0 = (x0 * a + c) % m;
    if (defaultVec)
        f << x0 % 1001 << ",";
    else
        f << x0 << ",";

    if (i == showsVector[check_progress_index]) {
        cout << (check_progress_index + 1) * 10 << "% completed\n";
        check_progress_index++;
    }
}
f.close();
return;
}

```

1.2 Аддитивный метод

Описание алгоритма:

Последовательность определяется следующим образом:

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

Инициализирующий вектор: модуль, младший индекс, старший индекс, последовательность начальных значений.

Параметры запуска программы:

*prng.exe/n:16000/g:add/i:8001,18,65,816,159,798,290,168,441,691,655,874,
220,125,977,586,381,868,294,948,437,581,181,701,536,11,672,103,601,794,189,12,130,
386,828,288,183,117,456,624,807,110,498,27,234,474,613,615,341,906,562,778,486,
155,276,894,441,226,762,234,762,98,458,399,445,765,223,879 /f:prngTestAdd.txt*

```
D:\cpp_projects\prng\prng>prng.exe /n:16000 /g:add /i:8001,18,65,816,159,798,290,168,441,691,655,874,220,125,977,586,381  
 ,868,294,948,437,581,181,701,536,11,672,103,601,794,189,12,130,386,828,288,183,117,456,624,807,110,498,27,234,474,613,61  
 5,341,906,562,778,486,155,276,894,441,226,762,234,762,98,458,399,445,765,223,879 /f:prngTestAdd.txt  
  
The process of generating pseudo-random numbers:  
  
10% completed  
20% completed  
30% completed  
40% completed  
50% completed  
60% completed  
70% completed  
80% completed  
90% completed  
100% completed  
  
Generation method: add  
Count of generated numbers: 16000  
Full name of the output file: prngTestAdd.txt
```

Рисунок 3 – Запуск программы

Рисунок 4 – Сгенерированная последовательность

Исходный код программы:

```
void addFunc(string codeMethod, vector <int> genVec, int NumCount, const char* fileName,
→ bool defaultVec) {
    // Input: module, low index, high index, sequence of initial values
    long long m = 8001;
    long long k = 18;
    long long i = 65;
```

```

long long x0;
vector <long long> xNs;
if (!defaultVec) {
    m = genVec[0];
    k = genVec[1];
    j = genVec[2];
    for (int i = 3; i < genVec.size(); i++)
    {
        xNs.push_back(genVec[i]);
    }
}
else {
    xNs = { 816, 159, 798, 290, 168, 441, 691, 655, 874, 220, 125, 977, 586,
    ↪ 381, 868, 294, 948, 437, 581, 181, 701, 536, 11, 672, 103, 601, 794,
    ↪ 189, 12, 130, 386, 828, 288, 183, 117, 456, 624, 807, 110, 498, 27, 234,
    ↪ 474, 613, 615, 341, 906, 562, 778, 486, 155, 276, 894, 441, 226, 762,
    ↪ 234, 762, 98, 458, 399, 445, 765, 223, 879 }
    ;
}
if (k < 1 || k >= j || j < 1) {
    cout << "\nError. The parameters do not meet the requirements.\n";
    return;
}

vector<int> showsVector = progressVector(NumCount);
int check_progress_index = 0;

ofstream f;
f.open(fileName, ios::out);
long long maxkj = max(k, j);
if (xNs.size() < maxkj) {
    cout << "\nError. The parameters do not meet the requirements (The
    ↪ transferred initial values are insufficient).\n";
    return;
}
for (int i = maxkj + 1; i < NumCount + maxkj + 1; i++)
{
    x0 = (xNs[i - k] + xNs[i - j]) % m;
    xNs.push_back(x0);
    if (defaultVec)
        f << x0 % 1001 << ",";
    else
        f << x0 << ",";
    if (i == showsVector[check_progress_index]) {
        cout << (check_progress_index + 1) * 10 << "% completed\n";
        check_progress_index++;
    }
}

```

```

        }
    }
    f.close();
    return;
}

```

1.3 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:

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

Инициализирующий вектор: p, q_1, q_2, q_3, w .

Параметры запуска программы:

`prng.exe /n:16000 /g:5p /i:4253,1093,2254,3297,16 /f:prngTest5p.txt`

```

D:\cpp_projects\prng\prng>prng.exe /n:16000 /g:5p /i:4253,1093,2254,3297,16 /f:prngTest5p.txt
The process of generating pseudo-random numbers:
10% completed
20% completed
30% completed
40% completed
50% completed
60% completed
70% completed
80% completed
90% completed
100% completed
Generation method: 5p
Count of generated numbers: 16000
Full name of the output file: prngTest5p.txt

```

Рисунок 5 – Запуск программы

<img alt="Screenshot of a Windows terminal window showing the generated sequence of numbers from the program. The sequence starts with 1, 429, 450, 961, 480, 976, 223, 847, 423, 211, 606, 803, 637, 318, 659, 329, 900, 450, 460, 730, 198, 295, 142, 306, 153, 577, 789, 630, 815, 833, 151, 311, 155, 813, 642, 821, 911, 691, 846, 658, 64, 32, 516, 493, 482, 977, 488, 744, 372, 421, 210, 195, 553, 276, 874, 937, 203, 337, 404, 702, 851, 425, 448, 224, 848, 924, 462, 231, 616, 43, 21, 511, 255, 127, 799, 635, 818, 409, 204, 838, 654, 62, 31, 516, 25, 8, 364, 918, 459, 730, 365, 918, 959, 715, 593, 197, 398, 434, 717, 859, 939, 266, 335, 903, 952, 211, 341, 906, 188, 329, 908, 185, 328, 399, 435, 718, 359, 915, 192, 596, 298, 384, 427, 949, 209, 104, 287, 879, 17, 4, 587, 28, 750, 119, 298, 645, 558, 14, 597, 253, 862, 666, 68, 534, 520, 486, 243, 857, 163, 817, 143, 572, 786, 893, 947, 709, 89, 44, 257, 364, 417, 944, 267, 839, 154, 312, 391, 696, 348, 374, 174, 87, 43, 757, 879, 4 39, 955, 212, 606, 803, 401, 936, 968, 984, 227, 614, 307, 654, 62, 266, 133, 567, 18, 244, 122, 61, 266, 633, 552, 276, 373, 687, 579, 790, 895, 447, 459, 465, 232, 616, 43, 522, 496, 984, 727, 599, 299, 650, 825, 6 48, 59, 538, 500, 750, 375, 688, 579, 525, 998, 734, 867, 433, 717, 859, 936, 965, 482, 241, 120, 796, 133, 302, 651, 325, 898, 684, 77, 539, 770, 128, 796, 133, 302, 151, 576, 788, 894, 682, 76, 38, 519, 259, 129, 5 65, 518, 494, 482, 741, 370, 921, 961, 981, 726, 863, 166, 819, 645, 823, 647, 559, 279, 139, 805, 983, 952, 711, 591, 796, 898, 449, 960, 715, 92, 46, 523, 261, 631, 816, 643, 557, 514, 757, 378, 189, 595, 798, 899 , 184, 592, 296, 148, 810, 140, 70, 535, 267, 133, 302, 151, 576, 288, 144, 888, 639, 820, 418, 941, 205, 838, 154, 813, 987, 689, 344, 172, 86, 779, 890, 945, 472, 236, 854, 662, 831, 415, 943, 266, 839, 419, 445, 7 23, 862, 666, 68, 34, 751, 612, 806, 993, 952, 476, 738, 604, 537, 504, 752, 376, 924, 962, 981, 726, 598, 534, 2, 1, 591, 496, 979, 489, 980, 725, 598, 299, 149, 310, 155, 813, 141, 79, 535, 583, 751, 611, 305, 388, 429, 214, 607, 539, 252, 862, 293, 445, 468, 978, 207, 654, 62, 266, 133, 567, 18, 244, 122, 61, 266, 633, 552, 276, 373, 687, 579, 790, 895, 447, 459, 465, 232, 616, 43, 522, 496, 984, 727, 599, 299, 650, 825, 6 2, 416, 944, 472, 471, 736, 868, 434, 717, 758, 414, 442, 721, 360, 916, 458, 729, 680, 535, 593, 987, 729, 865, 668, 69, 270, 635, 52, 26, 749, 109, 555, 513, 992, 731, 100, 285, 142, 807, 138, 569, 284, 878, 674, 3 37, 168, 310, 660, 836, 252, 862, 293, 445, 468, 978, 207, 654, 62, 266, 133, 567, 18, 244, 122, 61, 266, 633, 552, 276, 373, 687, 579, 790, 895, 447, 459, 465, 232, 616, 43, 522, 496, 984, 727, 599, 299, 650, 825, 6 171, 821, 145, 308, 654, 327, 664, 567, 18, 244, 122, 61, 266, 633, 552, 276, 373, 687, 579, 790, 895, 447, 459, 465, 232, 616, 43, 522, 496, 984, 727, 599, 299, 650, 825, 6 906, 953, 476, 974, 222, 111, 556, 278, 375, 938, 969, 494, 978, 224, 112, 556, 13, 242, 121, 68, 538, 0, 0, 235, 618, 389, 655, 327, 899, 685, 578, 524, 724, 381, 426, 213, 196, 53, 527, 999, 1000, 508, 250, 360, 316, 193, 832, 651, 826, 913, 9 57, 714, 92, 281, 876, 673, 572, 21, 246, 358, 414, 707, 353, 912, 191, 596, 33, 16, 243, 857, 163, 817, 408, 704, 587, 28, 514, 57, 629, 314, 657, 564, 782, 126, 298, 384, 692, 581, 791, 631, 50, 525, 763, 882, 176 , 88, 44, 257, 384, 557, 514, 257, 864, 167, 584, 292, 882, 676, 73, 537, 504, 752, 876, 673, 71, 771, 385, 428, 449, 960, 480, 248, 355, 913, 456, 964, 982, 226, 113, 792, 896, 683, 577, 23, 11, 5, 238, 855, 427, 949, 71 0, 208, 600, 535, 768, 119, 59, 538, 0, 736, 868, 934, 282, 681, 516, 3, 1, 236, 118, 294, 147, 73, 272, 872, 930, 968, 484, 242, 356, 914, 957, 478, 230, 855, 663, 331, 401, 436, 453, 962, 216, 108, 798, 130, 565, 28 2, 141, 571, 786, 628, 549, 775, 387, 193, 832, 916, 693, 582, 526, 494, 249, 625, 925, 713, 612, 541, 270, 871, 671, 79, 335, 2, 737, 368, 184, 92, 546, 773, 887, 443, 957, 213, 607, 303, 151, 811, 906, 53, 476, 974, 222, 111, 556, 278, 375, 938, 969, 494, 978, 224, 112, 556, 13, 242, 121, 68, 538, 0, 0, 235, 618, 389, 655, 327, 899, 685, 578, 524, 724, 381, 426, 213, 196, 53, 527, 999, 1000, 508, 250, 360, 316, 193, 832, 651, 826, 913, 9 57, 714, 92, 281, 876, 673, 572, 21, 246, 358, 414, 707, 353, 912, 191, 596, 33, 16, 243, 857, 163, 817, 408, 704, 587, 28, 514, 57, 629, 314, 657, 564, 782, 126, 298, 384, 692, 581, 791, 631, 50, 525, 763, 882, 176 , 88, 639, 319, 895, 648, 576, 723, 681, 593, 527, 999, 735, 868, 695, 934, 967, 719, 94, 547, 589, 496, 745, 873, 937, 468, 469, 978, 985, 492, 782, 491, 746, 873, 171, 321, 160, 315, 658, 329, 40 0, 208, 600, 535, 768, 119, 59, 538, 0, 736, 868, 934, 282, 681, 516, 3, 1, 236, 118, 294, 147, 73, 272, 872, 930, 968, 484, 242, 356, 914, 957, 478, 230, 855, 663, 331, 401, 436, 453, 962, 216, 108, 798, 130, 565, 28 2, 141, 571, 786, 628, 549, 775, 387, 193, 832, 916, 693, 582, 526, 494, 249, 625, 925, 713, 612, 541, 270, 871, 671, 79, 335, 2, 737, 368, 184, 92, 546, 773, 887, 443, 957, 213, 607, 303, 151, 811, 906, 53, 476, 974, 222, 111, 556, 278, 375, 938, 969, 494, 978, 224, 112, 556, 13, 242, 121, 68, 538, 0, 0, 235, 618, 389, 655, 327, 899, 685, 578, 524, 724, 381, 426, 213, 196, 53, 527, 999, 1000, 508, 250, 360, 316, 193, 832, 651, 826, 913, 9 57, 714, 92, 281, 876, 673, 572, 21, 246, 358, 414, 707, 353, 912, 191, 596, 33, 16, 243, 857, 163, 817, 408, 704, 587, 28, 514, 57, 629, 314, 657, 564, 782, 126, 298, 384, 692, 581, 791, 631, 50, 525, 763, 882, 176 , 88, 639, 319, 895, 648, 576, 723, 681, 593, 527, 999, 735, 868, 695, 934, 967, 719, 94, 547, 589, 496, 745, 873, 937, 468, 469, 978, 985, 492, 782, 491, 746, 873, 171, 321, 160, 315, 658, 329, 40 0, 208, 600, 535, 768, 119, 59, 538, 0, 736, 868, 934, 282, 681, 516, 3, 1, 236, 118, 294, 147, 73, 272, 872, 930, 968, 484, 242, 356, 914, 957, 478, 230, 855, 663, 331, 401, 436, 453, 962, 216, 108, 798, 130, 565, 28 2, 141, 571, 786, 628, 549, 775, 387, 193, 832, 916, 693, 582, 526, 494, 249, 625, 925, 713, 612, 541, 270, 871, 671, 79, 335, 2, 737, 368, 184, 92, 546, 773, 887, 443, 957, 213, 607, 303, 151, 811, 906, 53, 476, 974, 222, 111, 556, 278, 375, 938, 969, 494, 978, 224, 112, 556, 13, 242, 121, 68, 538, 0, 0, 235, 618, 389, 655, 327, 899, 685, 578, 524, 724, 381, 426, 213, 196, 53, 527, 999, 1000, 508, 250, 360, 316, 193, 832, 651, 826, 913, 9 57, 714, 92, 281, 876, 673, 572, 21, 246, 358, 414, 707, 353, 912, 191, 596, 33, 16, 243, 857, 163, 817, 408, 704, 587, 28, 514, 57, 629, 314, 657, 564, 782, 126, 298, 384, 692, 581, 791, 631, 50, 525, 763, 882, 176 , 88, 639, 319, 895, 648, 576, 723, 681, 593, 527, 999, 735, 868, 695, 934, 967, 719, 94, 547, 589, 496, 745, 873, 937, 468, 469, 978, 985, 492, 782, 491, 746, 873, 171, 321, 160, 315, 658, 329, 40 0, 208, 600, 535, 768, 119, 59, 538, 0, 736, 868, 934, 282, 681, 516, 3, 1, 236, 118, 294, 147, 73, 272, 872, 930, 968, 484, 242, 356, 914, 957, 478, 230, 855, 663, 331, 401, 436, 453, 962, 216, 108, 798, 130, 565, 28 2, 141, 571, 786, 628, 549, 775, 387, 193, 832, 916, 693, 582, 526, 494, 249, 625, 925, 713, 612, 541, 270, 871, 671, 79, 335, 2, 737, 368, 184, 92, 546, 773, 887, 443, 957, 213, 607, 303, 151, 811, 906, 53, 476, 974, 222, 111, 556, 278, 375, 938, 969, 494, 978, 224, 112, 556, 13, 242, 121, 68, 538, 0, 0, 235, 618, 389, 655, 327, 899, 685, 578, 524, 724, 381, 426, 213, 196, 53, 527, 999, 1000, 508, 250, 360, 316, 193, 832, 651, 826, 913, 9 57, 714, 92, 281, 876, 673, 572, 21, 246, 358, 414, 707, 353, 912, 191, 596, 33, 16, 243, 857, 163, 817, 408, 704, 587, 28, 514, 57, 629, 314, 657, 564, 782, 126, 298, 384, 692, 581, 791, 631, 50, 525, 763, 882, 176 , 88, 639, 319, 895, 648, 576, 723, 681, 593, 527, 999, 735, 868, 695, 934, 967, 719, 94, 547, 589, 496, 745, 873, 937, 468, 469, 978, 985, 492, 782, 491, 746, 873, 171, 321, 160, 315, 658, 329, 40 0, 208, 600, 535, 768, 119, 59, 538, 0, 736, 868, 934, 282, 681, 516, 3, 1, 236, 118, 294, 147, 73, 272, 872, 930, 968, 484, 242, 356, 914, 957, 478, 230, 855, 663, 331, 401, 436, 453, 962, 216, 108, 798, 130, 565, 28 2, 141, 571, 786, 628, 549, 775, 387, 193, 832, 916, 693, 582, 526, 494, 249, 625, 925, 713, 612, 541, 270, 871, 671, 79, 335, 2, 737, 368, 184, 92, 546, 773, 887, 443, 957, 213, 607, 303, 151, 811, 906, 53, 476, 974, 222, 111, 556, 278, 375, 938, 969, 494, 978, 224, 112, 556, 13, 242, 121, 68, 538, 0, 0, 235, 618, 389, 655, 327, 899, 685, 578, 524, 724, 381, 426, 213, 196, 53, 527, 999, 1000, 508, 250, 360, 316, 193, 832, 651, 826, 913, 9 57, 714, 92, 281, 876, 673, 572, 21, 246, 358, 414, 707, 353, 912, 191, 596, 33, 16, 243, 857, 163, 817, 408, 704, 587, 28, 514, 57, 629, 314, 657, 564, 782, 126, 298, 384, 692, 581, 791, 631, 50, 525, 763, 882, 176 , 88, 639, 319, 895, 648, 576, 723, 681, 593, 527, 999, 735, 868, 695, 934, 967, 719, 94, 547, 589, 496, 745, 873, 937, 468, 469, 978, 985, 492, 782, 491, 746, 873, 171, 321, 160, 315, 658, 329, 40 0, 208, 600, 535, 768, 119, 59, 538, 0, 736, 868, 934, 282, 681, 516, 3, 1, 236, 118, 294, 147, 73, 272, 872, 930, 968, 484, 242, 356, 914, 957, 478, 230, 855, 663, 331, 401, 436, 453, 962, 216, 108, 798, 130, 565, 28 2, 141, 571, 786, 628, 549, 775, 387, 193, 832, 916, 693, 582, 526, 494, 249, 625, 925, 713, 612, 541, 270, 871, 671, 79, 335, 2, 737, 368, 184, 92, 546, 773, 887, 443, 957, 213, 607, 303, 151, 811, 906, 53, 476, 974, 222, 111, 556, 278, 375, 938, 969, 494, 978, 224, 112, 556, 13, 242, 121, 68, 538, 0, 0, 235, 618, 389, 655, 327, 899, 685, 578, 524, 724, 381, 426, 213, 196, 53, 527, 999, 1000, 508, 250, 360, 316, 193, 832, 651, 826, 913, 9 57, 714, 92, 281, 876, 673, 572, 21, 246, 358, 414, 707, 353, 912, 191, 596, 33, 16, 243, 857, 163, 817, 408, 704, 587, 28, 514, 57, 629, 314, 657, 564, 782, 126, 298, 384, 692, 581, 791, 631, 50, 525, 763, 882, 176 , 88, 639, 319, 895, 648, 576, 723, 681, 593, 527, 999, 735, 868, 695, 934, 967, 719, 94, 547, 589, 496, 745, 873, 937, 468, 469, 978, 985, 492, 782, 491, 746, 873, 171, 321, 160, 315, 658, 329, 40 0, 208, 600, 535, 768, 119, 59, 538, 0, 736, 868, 934, 282, 681, 516, 3, 1, 236, 118, 294, 147, 73, 272, 872, 930, 968, 484, 242, 356, 914, 957, 478, 230, 855, 663, 331, 401, 436, 453, 962, 216, 108, 798, 130, 565, 28 2, 141, 571, 786, 628, 549, 775, 387, 193, 832, 916, 693, 582, 526, 494, 249, 625, 925, 713, 612, 541, 270, 871, 671, 79, 335, 2, 737, 368, 184, 92, 546, 773, 887, 443, 957, 213, 607, 303, 151, 811, 906, 53, 476, 974, 222, 111, 556, 278, 375, 938, 969, 494, 978, 224, 112, 556, 13, 242, 121, 68, 538, 0, 0, 235, 618, 389, 655, 327, 899, 685, 578, 524, 724, 381, 426, 213, 196, 53, 527, 999, 1000, 508, 250, 360, 316, 193, 832, 651, 826, 913, 9 57, 714, 92, 281, 876, 673, 572, 21, 246, 358, 414, 707, 353, 912, 191, 596, 33, 16, 243, 857, 163, 817, 408, 704, 587, 28, 514, 57, 629, 314, 657, 564, 782, 126, 298, 384, 692, 581, 791, 631, 50, 525, 763, 882, 176 , 88, 639, 319, 895, 648, 576, 723, 681, 593, 527, 999, 735, 868, 695, 934, 967, 719, 94, 547, 589, 496, 745, 873, 937, 468, 469, 978, 985, 492, 782, 491, 746, 873, 171, 321, 160, 315, 658, 329, 40 0, 208, 600, 535, 768, 119, 59, 538, 0, 736, 868, 934, 282, 681, 516, 3, 1, 236, 118, 294, 147, 73, 272, 872, 930, 968, 484, 242, 356, 914, 957, 478, 230, 855, 663

```

ofstream f;
f.open(fileName, ios::out);
for (int i = 0; i < NumCount; i++)
{
    bool xNplusP = reg[regLength - q1 - 1] ^ reg[regLength - q2 - 1] ^
    ↵ reg[regLength - q3 - 1];
    reg.push_front(xNplusP);
    reg.pop_back();
    long long ReginDec = 0;
    for (int h = 0; h < w; h++)
    {
        ReginDec *= 2;
        ReginDec += reg[h];
    }
    if (defaultVec)
        f << ReginDec % 1001 << ",";
    else
        f << ReginDec << ",";
    if (i == showsVector[check_progress_index]) {
        cout << (check_progress_index + 1) * 10 << "% completed\n";
        check_progress_index++;
    }
}
f.close();
return;
}

```

1.4 Регистр сдвига с обратной связью (РСЛОС)

Описание алгоритма:

Для натурального числа p и $c_1, c_2, \dots, c_{(p-1)}$, принимающих значения 0 или 1 определяют рекуррентную формулу

$$X_{n+p} = a_{p-1}X_{n+p-1} + a_{p-2}X_{n+p-2} + \dots + a_1X_{n+1} + X_n$$

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, \dots, a_{p-1} .
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.

Инициализирующий вектор: двоичное представление вектора коэффициентов, начальное значение регистра.

Параметры запуска программы:

prng.exe /n:16000 /g:lfsr /i:100000000101011,121 /f:prngTestLfsr.txt

```
D:\cpp_projects\prng>prng.exe /n:16000 /g:lfsr /i:100000000101011,121 /f:prngTestLfsr.txt
The process of generating pseudo-random numbers:

10% completed
20% completed
30% completed
40% completed
50% completed
60% completed
70% completed
80% completed
90% completed
100% completed

Generation method: lfsr
Count of generated numbers: 16000
Full name of the output file: prngTestLfsr.txt
```

Рисунок 7 – Запуск программы

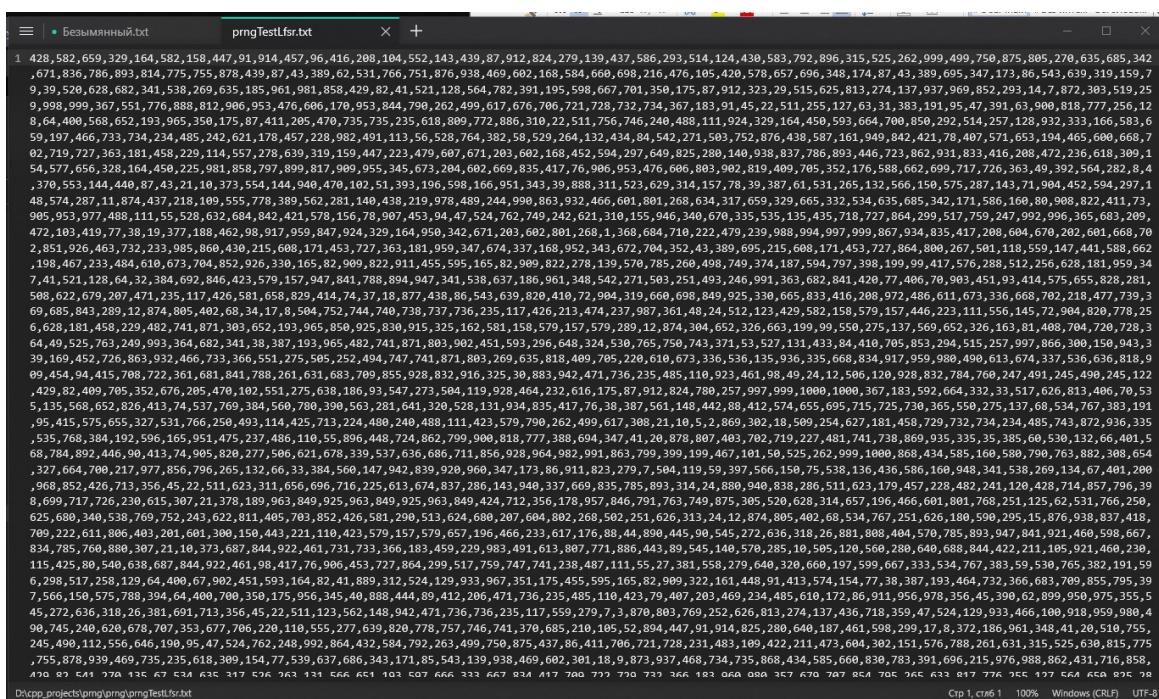


Рисунок 8 – Сгенерированная последовательность

Исходный код программы:

```
void lfsrFunc(string codeMethod, vector<int> genVec, int NumCount, const char* fileName,
← bool defaultVec) {
    // Input: binary representation of the coefficient vector, initial value of the
    ← register
    if (defaultVec) {
        genVec = { 121, 14, 5, 3, 1, 0 }; // x0, coeffs (100000000101011)
    }
}
```

```

int x0 = genVec[0];
int x0Length = log2(x0) + 1;
int regLength = x0Length - 1;
bool bigCoeff = false;
for (int i = 1; i < genVec.size(); i++)
{
    if (genVec[i] > regLength) {
        regLength = genVec[i] + 1;
        bigCoeff = true;
    }
}
if (x0 < 0) {
    cout << "\nError. The parameters do not meet the requirements.\n";
    return;
}

vector<int> showsVector = progressVector(NumCount);
int check_progress_index = 0;

deque<bool> reg;
if (!bigCoeff) {
    regLength++;
    reg = binary(x0, regLength);
}
else {
    reg.assign(regLength - x0Length, 0);
    deque<bool> regDop = binary(x0, x0Length);
    for (int i = 0; i < regDop.size(); i++)
    {
        reg.push_back(regDop[i]);
    }
}
ofstream f;
f.open(fileName, ios::out);
for (int i = 0; i < NumCount; i++)
{
    bool xNplusP = reg[regLength - genVec[i] - 1];
    for (int j = 2; j < genVec.size(); j++)
    {
        xNplusP ^= reg[regLength - genVec[j] - 1];
    }
    reg.push_front(xNplusP);
    reg.pop_back();
    long long ReginDec = 0;
    for (int h = 0; h < reg.size(); h++)
    {
        ReginDec *= 2;
        ReginDec += reg[h];
    }
}

```

```

    }

    if (defaultVec)
        f << ReginDec % 1001 << ",";
    else
        f << ReginDec << ",";

    if (i == showsVector[check_progress_index]) {
        cout << (check_progress_index + 1) * 10 << "% completed\n";
        check_progress_index++;
    }
}

f.close();
return;
}

```

1.5 Нелинейная комбинация РСЛОС

Описание алгоритма:

Последовательность получается объединением нелинейным образом трёх РСЛОС. Нелинейная функция генератора:

$$f(x_1, x_2, x_3) = x_1x_2 \oplus (1 + x_2)x_3 = x_1x_2 \oplus x_2x_3 \oplus x_3$$

Инициализирующий вектор: двоичное представление векторов коэффициентов для $R1, R2, R3$.

Параметры запуска программы:

`prng.exe /n:16000 /g:nfsr /i:100000000011011,100000000000011,
10000000000101101 /f:prngTestNfsr.txt`

```

D:\cpp_projects\prng>prng.exe /n:16000 /g:nfsr /i:100000000011011,100000000000011,1000000000101101 /f:prngTestNfsr.txt

The process of generating pseudo-random numbers:

10% completed
20% completed
30% completed
40% completed
50% completed
60% completed
70% completed
80% completed
90% completed
100% completed

Generation method: nfsr
Count of generated numbers: 16000
Full name of the output file: prngTestNfsr.txt

```

Рисунок 9 – Запуск программы

Безымянный.txt

```
prngTestNfsr.txt
```

```
1 228,585,292,646,323,62,316,158,79,540,278,135,568,755,878,439,219,580,260,601,271,636,288,144,72,36,18,480,711,856,398,670,335,668,394,652,826,913,957,949,975,458,229,85,4
2,992,967,954,977,459,730,335,668,895,373,157,78,510,755,348,144,72,536,768,884,913,456,699,828,410,705,352,676,809,875,408,174,57,529,235,88,515,728,835,888,414,287,74,37,
519,736,335,167,83,512,256,98,19,488,210,605,362,651,826,913,957,449,695,347,144,543,742,371,656,828,384,162,81,541,771,356,178,568,789,368,180,590,795,397,169,555,248,94,1
7,988,961,451,225,83,41,521,731,866,904,452,226,83,512,226,83,512,756,878,939,970,985,463,262,601,801,490,290,180,26,981,490,215,578,289,645,293,647,294,117,529,765,883,412
,286,574,787,864,993,922,961,451,194,68,34,517,729,364,653,297,141,545,743,873,936,968,454,227,113,527,263,102,522,231,86,543,742,842,391,666,884,873,937,468,705,323,632,31
6,128,564,782,361,681,811,876,488,704,322,661,831,916,958,979,489,745,372,657,299,626,310,655,327,163,582,261,631,816,408,204,102,522,261,631,315,628,284,142,71,6,474,237,5
89,795,898,920,931,966,453,197,599,270,105,52,26,484,742,871,936,964,484,713,356,148,545,277,282,524,232,760,779,890,415,708,354,648,795,368,154,548,774,357,679,339,169,84
,513,256,128,564,282,141,541,741,370,685,342,171,556,278,610,776,388,164,82,41,992,496,719,830,385,693,317,129,35,488,715,357,178,89,44,493,217,579,790,895,447,694,347,644,2
92,646,293,617,809,375,688,344,672,306,123,33,487,214,77,9,4,973,457,228,614,397,624,783,892,446,193,567,754,347,173,86,13,978,459,788,821,911,426,183,91,546,273,607,303,62
2,811,966,453,697,819,489,204,72,587,724,332,637,289,115,57,529,735,338,169,55,498,749,875,938,439,690,215,528,814,188,94,518,759,858,896,418,179,68,38,15,478,739,369,5
84,92,546,744,843,892,446,193,567,283,612,866,874,437,689,845,923,432,216,188,554,777,859,406,708,858,896,948,945,472,236,618,180,396,165,583,262,131,36,989,494,718,830,915
,958,449,195,68,585,252,597,298,628,280,611,305,623,282,111,55,528,264,182,275,122,561,751,871,936,439,690,816,879,940,440,691,841,894,417,679,318,626,31
3,156,78,39,490,745,372,156,549,274,107,554,247,594,797,869,905,534,447,194,97,48,24,483,241,120,531,736,839,419,680,811,405,673,307,624,282,612,777,889,444,222,111,55,27,4
84,713,356,148,7,4,74,474,207,103,522,231,616,308,154,548,243,622,211,616,305,252,126,534,237,589,265,633,817,879,910,425,683,341,641,791,896,919,960,480,210,605,773,386,193,597,769,384,621,316,29
,314,657,829,414,678,818,905,953,947,444,693,847,423,712,827,884,412,209,574,287,114,27,13,507,754,87
7,909,925,433,717,859,429,214,687,274,107,94,243,983,462,481,211,608,303,151,546,243,624,222,311,126,63,236,88,544,723,886,443,692,817,909,425,683,341,141,571,285,643,292,116,2
8,14,478,239,99,548,743,423,671,335,668,394,612,812,486,783,251,497,228,614,396,762,331,666,884,873,497,674,237,639,798,365,182,591,766,383,1
91,566,783,892,446,193,96,548,244,122,561,751,375,158,79,39,991,996,468,795,323,662,831,386,193,96,519,230,615,805,684,904,923,432,186,93,46,994,997,469,735,367,684,813,877,999
,425,713,356,148,248,184,542,354,677,839,890,916,458,229,585,292,116,28,485,213,77,509,225,583,291,211,646,794,367,183,62,502,221,611,776,149,545,743,371,656,828,414,768,83
8,989,445,222,611,776,388,164,52,526,763,381,691,345,143,42,521,761,851,396,669,380,623,282,111,55,528,263,632,787,894,918,429,214,107,554,247,594,267,684,272,636,288,644,793,
367,654,798,369,655,234,298,149,575,758,349,174,587,293,117,529,765,853,897,919,938,465,953,476,238,598,265,603,272,136,530,269,605,803,872,936,968,984,492,746,373,687,814,907,
453,226,584,792,867,433,717,829,915,958,479,749,841,391,696,618,638,786,864,932,466,733,337,669,885,983,942,221,76,538,239,628,280,640,299,215,28,485,213,106,524,762,852,92
6,934,967,454,722,334,137,569,755,848,924,962,481,461,211,608,303,151,546,243,624,222,311,126,63,236,88,544,723,886,443,692,817,909,425,683,341,141,571,285,643,292,116,2
4,627,284,112,527,734,337,139,112,491,716,358,149,74,7,474,708,354,177,559,758,345,643,792,366,683,341,179,556,248,124,562,251,125,563,782,368,651,791,894,419,688,811,996,9
4,432,716,829,915,928,935,968,484,742,871,966,453,226,613,777,359,179,89,15,478,710,325,633,817,379,160,80,176,238,590,265,132,66,53,487,714,857,428,184,563,281,611,276
,638,289,191,756,258,294,618,719,139,569,756,878,409,175,514,257,599,178,442,191,66,3,1,972,957,949,445,222,611,276,108,525,725,733,366,183,592,796,898,419,710,325,16
2,51,25,513,757,849,424,182,61,531,766,854,898,920,986,951,475,738,339,149,541,241,591,266,183,51,526,263,131,65,32,16,508,254,127,534,237,589,765,382,161,551,776,388,194,9
7,549,775,888,915,958,479,718,855,287,684,813,877,919,938,465,953,476,238,598,265,603,272,136,530,269,605,803,872,936,968,984,447,746,373,687,814,907,
5,293,235,117,58,500,220,581,298,645,293,146,544,242,121,531,736,868,404,792,351,676,838,389,194,568,755,848,895,447,694,818,999,454,227,584,279,211,70,110,25,513,727,363,181
,561,251,596,268,605,273,107,524,262,181,50,525,262,131,65,533,767,854,397,669,805,873,436,718,859,930,435,718,359,688,340,641,821,911,926,433,216,78,9,505,223,82,41,491,74
6,844,893,947,473,737,339,169,555,624,284,622,612,806,901,422,181,61,501,721,331,136,539,740,841,420,180,561,751,375,688,344,142,41,491,716,288,134,37,193,509,725,863,431,68
6,313,127,564,282,111,526,233,587,293,617,308,625,813,406,203,602,801,871,435,718,359,650,825,412,706,824,912,456,198,99,558,746,844,422,181,61,1,501,721,331,136,68,534,738
,369,184,194,527,158,99,558,257,587,664,332,666,383,151,462,482,712,827,384,692,346,143,42,492,246,623,282,612,886,874,407,704,322,661,891,400,260,108,559,245,122,61,1,0,9
71,456,728,864,903,451,726,333,667,384,623,812,906,924,933,937,439,690,849,393,697,348,645,293,617,279,610,895,903,952,446,193,96,548,774,387,193,597,799,878,405,202,71,986
,724,362,652,296,118,59,530,265,633,99,529,235,949,474,875,908,447,694,818,999,454,227,584,279,211,70,110,25,513,727,363,181
5,307,654,327,163,522,246,594,768,855,427,744,327,664,332,166,83,542,271,106,53,527,734,337,168,54,527,234,117,529,765,883,942,942,942,471,235,618,708,890,945,443,221,81,40
,20,981,461,731,336,138,39,991,466,233,116,529,235,117,559,279,640,791,395,668,334,137,539,269,605,302,151,46,94,467,734,838,898,916,958,950,445,193,96,548,274,637,819,910
,955,477,789,855,427,213,577,789,394,167,554,277,639,820,410,285,73,7,594,222,582,762,381,196,95,18,988,468,230,115,28,14,478,739,848,420,710,325,633,287,644,322,661,891,40
0,671,306,653,297,148,74,537,769,885,442,191,566,754,848,895,918,930,965,482,241,591,796,398,199,70,35,518,229,585,763,852,926,433,687,314,628,314,127,34,488,244,622,311,65
6 708 600 224 122 6 3 1 0 711 084 001 067 047 707 200 75 528 760 255 148 574 387 142 47 21 10 5 074 087 064 053 047 472 717 220 688 204 415 178 559 741 846 422
```

Cr1, Cr6 1 100% Windows (CRF) UTF-8

Рисунок 10 – Сгенерированная последовательность

Исходный код программы:

```
void nfsrFunc(string codeMethod, vector <int> genVec, int NumCount, const char* fileName,
    vector <deque <bool>> nfsrRegs, bool defaultVec) {
    // Input: binary representation of the coefficient vectors for R1, R2, R3
    if (defaultVec) {
        genVec = { 5, 14, 4, 3, 1, 0, 3, 15, 1, 0, 5, 16, 5, 3, 2, 0 }; // size1,coeffs1,size2,coeffs2,size3,coeffs3
        nfsrRegs = { {1,0,0,0,0,0,0,0,0,0,0,1,1,0,1,1}, {1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1}, {1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1} };
    }
    int nachR1 = 1;
    int endR1 = genVec[0] + 1;
    int nachR2 = endR1 + 1;
    int endR2 = genVec[endR1] + nachR2 + 1;
    int nachR3 = endR2 + 1;
    int endR3 = genVec.size();
    vector<int> startRslos = { nachR1, nachR2, nachR3 };
    vector<int> endssRslos = { endR1, endR2, endR3 };
    vector<int> regLengths;
    for (int i = 0; i < 3; i++)
    {
        regLengths.push_back(nfsrRegs[i].size());
    }
    vector <deque<bool>> regs = nfsrRegs;
```

```

vector<int> showsVector = progressVector(NumCount);
int check_progress_index = 0;

ofstream f;
f.open(fileName, ios::out);
for (int i = 0; i < NumCount; i++)
{
    for (int j = 0; j < 3; j++)
    {
        int startH = startsRslos[j];
        int endH = endssRslos[j];
        bool xNext = regs[j][regLengths[j] - genVec[startH] - 1];
        for (int h = startH; h < endH; h++)
        {
            xNext ^= regs[j][regLengths[j] - genVec[h] - 1];
        }
        regs[j].push_front(xNext);
        regs[j].pop_back();
    }
    vector<deque<bool>> regs2(2);
    regs2[0] = multRs(regs[0], regs[1]);
    regs2[1] = multRs(regs[1], regs[2]);
    deque<bool> resR = xorRs(xorRs(regs2[0], regs2[1]), regs[2]);

    long long ReginDec = 0;
    for (int h = 0; h < resR.size(); h++)
    {
        ReginDec *= 2;
        ReginDec += resR[h];
    }
    if (defaultVec)
        f << ReginDec % 1001 << ",";
    else
        f << ReginDec << ",";

    if (i == showsVector[check_progress_index]) {
        cout << (check_progress_index + 1) * 10 << "% completed\n";
        check_progress_index++;
    }
}
f.close();
return;
}

```

1.6 Вихрь Мерсенна

Описание алгоритма:

Метод Вихрь Мерсенна позволяет генерировать последовательность дво-

ичных псевдослучайных целых w-битных чисел в соответствии с 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 реккуррентности, $1 \leq q \leq p$;

X_n - w -битное 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 а;

XA – произведение, при вычислении которого сначала выполняют операцию $X >> 1$, если последний бит X равен 0, а затем, когда последний бит X равен 1, то вычисляют $XA = (X >> 1) \oplus A$

Инициализирующий вектор: модуль, начальное значение x .

Параметры запуска программы:

`prng.exe /n:16000 /g:mt /i:10001,8191 /f:prngTestMt.txt`

```
D:\cpp_projects\prng>prng.exe /n:16000 /g:mt /i:10001,8191 /f:prngTestMt.txt
The process of generating pseudo-random numbers:
10% completed
20% completed
30% completed
40% completed
50% completed
60% completed
70% completed
80% completed
90% completed
100% completed
Generation method: mt
Count of generated numbers: 16000
Full name of the output file: prngTestMt.txt
```

Рисунок 11 – Запуск программы

Рисунок 12 – Сгенерированная последовательность

Исходный код программы:

```
void mtFunc(string codeMethod, vector <int> genVec, int NumCount, const char* fileName, bool
↪ defaultVec) {
    // Input: module, initial value of x
    int p = 624;
    int w = 32;
    //string uBin = "1000000000000000000000000000000000000000000000000000000000000000";
    //string hBin = "0111111111111111111111111111111111111111111111111111111111111111";
    long long uDec = 2147483647;
    long long hDec = 2147483648;
    long long r = 31;
    long long q = 397;
    long long a = 2567483615;
    long long u = 11;
    long long s = 7;
    long long t = 15;
    long long l = 18;
    long long b = 2636928640;
    long long c = 4022730752;
    if (defaultVec) {
        genVec = { 10001, 8191 };
    }
    long long zMod = genVec[0];
    long long x0 = genVec[1];
    vector <long long> x_zns(1, x0);
    for (int i = 0; i < p; i++)
    {
        long long x = (x0 * u + q) % zMod;
        genVec.push_back(x);
        x0 = x;
    }
}
```

```

        x_zns.push_back(abs(1812433253 * (x_zns[i - 1] ^ (x_zns[i - 1] >> 30)) +
        ↵ i));
    }

vector<int> showsVector = progressVector(NumCount);
int check_progress_index = 0;

ofstream f;
f.open(fileName, ios::out);
int n = 0;
for (int countN = 0; countN < NumCount; countN++)
{
    long long xn = x_zns[n];
    long long xnPlus1 = x_zns[(n + 1) % p];
    long long yNew = (xn & uDec) | (xnPlus1 & hDec);
    int yNewLength = log2(yNew) + 1;
    deque<bool> yBin = binary(yNew, yNewLength);
    long long Xnew;
    yNew >>= 1;
    long long xnqp = x_zns[(n + q) % p];
    if (yBin[yBin.size() - 1] == 1)
        Xnew = xnqp ^ yNew ^ a;
    else
        Xnew = xnqp ^ yNew ^ 0;
    yNew = Xnew;
    yNew = yNew ^ (yNew >> u);
    yNew = yNew ^ ((yNew << s) & b);
    yNew = yNew ^ ((yNew << t) & c);
    long long zNew = yNew ^ (yNew >> l);
    if (defaultVec)
        f << (zNew % zMod) % 1001 << ",";
    else
        f << zNew % zMod << ",";
    if (countN == showsVector[check_progress_index]) {
        cout << (check_progress_index + 1) * 10 << "% completed\n";
        check_progress_index++;
    }
    x_zns[n] = Xnew;
    n = (n + 1) % p;
}
f.close();
return;
}

```

1.7 RC4

Описание алгоритма:

1. Инициализация $S_i, i = 0, 1, \dots, 255$.
 - a) for $i = 0$ to 255 : $S_i = i$
 - б) $j = 0$;
 - в) 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:
 - а) $i = (i + 1) \bmod 256$;
 - б) $j = (j + S_i) \bmod 256$;
 - в) $Swap(S_i, S_j)$;
 - г) $t = (S_i + S_j) \bmod 256$;
 - д) $K = S_t$.

Инициализирующий вектор: 256 начальных значений.

Параметры запуска программы:

```
prng.exe/n:16000/g:rc4/i:802,720,341,337,961,882,417,785,198,727,899,372,
374,425,556,615,813,768,840,183,893,568,73,387,18,436,182,125,806,899,485,607,
619,825,944,579,707,360,363,904,87,262,276,460,687,831,75,499,599,915,681,492,
483,754,878,500,189,60,624,994,959,109,600,577,934,544,156,640,903,519,544,990,
781,819,449,468,650,524,967,248,438,647,739,920,400,617,419,588,676,43,581,634,
151,181,211,84,724,367,723,627,886,267,617,667,85,65,134,735,589,100,983,26,747,
721,945,147,337,364,734,13,406,315,647,556,496,858,640,220,224,362,847,110,629,
463,776,713,528,909,448,116,9,430,141,755,151,86,901,488,449,635,500,855,950,147,
410,446,4,49,665,227,411,511,336,39,974,112,752,501,21,200,617,29,629,757,784,779,
843,684,266,292,319,766,146,269,912,556,714,916,605,378,142,15,889,478,54,862,590,
806,363,610,5,979,638,634,736,421,413,578,105,679,869,424,444,14,692,356,569,405,
271,173,783,413,188,671,891,242,533,480,48,895,89,53,873,727,686,608,147,98,185,
252,776,54,675,220,67,366,576,636,771,846,808,553,259,996,224,149/f:prngTestRc4.txt
```

```
D:\cpp_projects\prng\prng>prng.exe /n:16000 /g:rc4 /i:802,720,341,337,961,882,417,785,198,727,899,372,374,425,556,615,81  
3,768,840,183,893,568,73,387,18,436,182,125,806,899,485,607,619,825,944,579,707,360,363,904,87,262,276,460,687,831,75,49  
9,599,915,681,492,483,754,878,500,189,60,624,994,959,109,606,577,934,544,156,640,903,519,544,990,781,819,449,468,658,524  
,967,248,438,647,739,920,400,617,419,588,676,43,581,634,151,181,211,84,724,367,723,627,886,267,617,667,85,65,134,735,589  
,100,983,26,747,721,945,147,337,364,734,13,406,315,647,556,496,858,640,220,224,362,847,110,629,463,776,713,528,909,448,1  
16,9438,755,151,86,901,488,449,635,500,855,950,147,410,446,4,49,665,227,411,511,336,39,974,112,752,501,21,200,617,2  
9,629,757,784,779,843,684,266,292,319,766,146,269,912,556,714,916,605,378,142,15,889,478,54,862,590,806,363,610,5,979,63  
8,634,736,421,413,578,105,679,869,424,444,14,692,356,569,405,271,173,783,413,188,671,891,242,533,480,48,895,89,53,873,72  
7,686,688,147,98,185,252,776,54,675,220,67,366,576,636,771,846,808,553,259,996,224,149 /f:prngTestRc4.txt
```

Рисунок 13 – Запуск программы

● Безымянный.txt prngTestRc4.txt +

```
1 617,889,846,226,967,556,436,568,185,26,727,895,5,556,598,576,598,579,589,605,915,713,21,736,629,724,776,728,405,65,43,589,787,336,49,888,438,721,727,18,48,886,189,15,81  
9,825,26,405,667,634,15,528,43,378,43,483,862,262,141,847,427,847,886,421,629,961,188,488,783,224,336,419,983,886,647,29,378,629,189,259,882,888,568,912,776,739,492,533,142  
,797,9,478,686,411,364,607,577,783,754,692,688,944,21,608,735,665,208,385,459,441,619,886,449,692,259,843,146,681,556,411,771,926,847,783,617,341,983,962,739,847,110  
,421,417,501,149,113,619,107,160,188,882,460,29,634,676,448,528,676,681,757,599,24,573,24,587,24,586,858,387,496,259,886,444,468,624,306,424,280,112,979,492,996,  
920,98,374,200,754,736,85,363,406,496,819,613,469,501,147,577,617,364,549,627,777,635,478,647,544,501,698,198,382,172,584,721,916,43,54,784,634,544,893,692,259,408,478,  
147,889,577,444,959,934,43,736,211,24,616,813,48,372,15,376,378,257,21,444,367,200,15,727,100,198,984,667,249,177,349,736,300,186,792,979,449,292,889,768,444,688,721,245,895,114,  
873,142,723,53,916,425,901,367,776,776,188,364,994,424,565,556,116,108,224,463,14,945,617,776,843,993,556,116,785,411,183,724,904,734,771,146,226,367,619,142,424,125,783,189,739,858,42  
4,687,524,183,889,39,364,605,635,899,9,996,362,53,147,934,580,319,755,647,640,888,528,542,578,364,891,781,754,533,524,463,492,553,996,374,634,556,638,188,146,259,485,495,43  
0,569,511,362,757,825,141,967,524,687,336,598,792,141,292,629,806,901,755,668,436,779,681,438,134,862,39,979,999,436,810,916,337,48,500,13,75,29,151,553,337,372,185,855,2  
62,228,75,266,858,367,599,446,488,391,852,862,319,734,589,934,384,43,692,444,363,181,636,156,183,981,976,336,29,87,819,862,787,617,24,779,813,265,492,524,981,831,341,6  
17,48,727,858,356,181,378,485,959,647,776,687,54,151,478,971,95,768,446,996,598,104,635,893,98,824,142,983,893,610,916,556,315,615,460,974,141,316,198,735,650,721,114,008,134,  
480,363,640,341,893,405,425,636,771,553,891,992,196,171,991,807,481,769,146,45,430,228,271,617,510,505,476,768,806,220,819,449,671,873,606,807,525,951,151,173,5,727,267,727,544,220,  
68,608,436,600,544,147,998,889,961,413,259,915,14,68,367,752,438,577,967,607,987,634,319,248,496,108,889,356,500,647,53,81,85,808,665,871,151,819,543,588,424,188,224,259,  
85,721,142,757,91,901,366,125,727,619,698,944,267,675,271,686,988,388,125,413,87,580,886,319,571,974,774,408,48,889,806,524,575,544,959,916,26,588,519,776,903,508,508,819,410,276,  
79,73,608,151,39,813,718,966,142,599,198,873,468,118,492,173,636,98,781,181,488,992,636,84,146,499,891,881,211,189,151,996,979,916,528,723,714,579,595,599,878,417,589,  
899,929,134,959,617,4,478,86,372,185,785,617,752,336,173,994,372,855,369,242,747,276,766,496,146,901,488,636,524,227,785,269,757,480,617,627,647,485,511,801,736,478,266,  
3,695,862,341,227,713,617,987,723,899,438,619,687,568,945,192,208,173,785,598,721,266,425,499,15,25,679,106,185,483,776,888,327,846,535,577,752,60,410,48,26,636,727,684,723,1  
51,499,519,266,211,228,912,781,581,188,776,893,543,599,436,242,147,65,899,444,110,362,766,271,590,151,886,569,181,397,762,638,579,615,151,819,556,899,399,556,974,  
13,755,149,413,869,424,869,211,9,185,723,624,776,936,337,617,608,495,449,374,394,568,341,843,776,488,43,727,182,77,187,947,147,891,151,319,299,403,87,569,480,629,189,556,776,899  
892,276,67,869,878,813,561,754,334,367,156,116,439,148,378,488,813,619,629,799,436,367,271,273,857,862,676,13,528,848,686,508,543,194,584,578,363,220,707,784,86,797,367,424,  
713,252,590,779,533,146,421,466,156,784,849,399,259,886,49,895,389,159,567,847,182,928,776,224,97,85,151,341,54,436,424,584,396,228,576,776,671,891,766,94,634,5,142,721,71,6  
48,337,142,147,766,967,446,544,606,957,581,889,185,356,134,556,408,781,619,689,5,752,785,499,483,242,619,468,776,417,189,544,588,360,640,647,269,492,75,374,671,266,84,337,98,  
1,899,228,500,336,713,687,15,544,785,492,252,508,581,665,147,647,449,779,198,819,994,959,727,292,227,695,405,647,695,915,792,134,181,899,100,776,781,679,687,77,75,959,18,  
889,366,734,91,727,967,967,422,569,363,496,516,417,474,421,183,675,411,463,982,781,771,825,226,999,61,96,679,511,524,528,298,206,213,983,568,508,319,313,634,  
638,87,886,220,779,60,483,813,485,488,624,413,889,528,847,846,752,556,9,999,727,617,442,924,808,947,383,367,476,436,417,862,374,627,53,629,147,581,933,999,292,2  
04,692,846,825,556,449,729,530,151,483,650,735,886,724,647,267,519,424,617,266,896,912,776,551,362,784,994,276,498,529,226,598,569,887,898,446,364,733,831,5,934,65,528,367,  
75,843,372,556,799,735,544,912,747,882,171,573,597,617,337,695,886,608,579,453,611,471,727,181,265,993,635,417,39,634,969,724,752,15,528,619,840,259,934,569,688,463,2  
1,915,889,500,200,899,399,364,720,911,617,741,496,15,524,590,374,511,158,689,599,916,794,617,889,319,686,441,648,540,511,783,500,862,889,944,984,629,182,372,747,675,776,551  
893,629,149,934,974,768,198,488,363,424,114,896,519,485,285,341,605,438,553,916,67,524,341,728,15,211,119,413,269,467,740,595,599,588,785,912,466,776,687,605,73,419,5  
56,108,499,945,681,707,899,4,869,499,366,276,647,468,576,485,425,372,410,855,496,945,556,843,739,98,636,617,501,675,576,149,967,766,252,714,227,634,617,488,950,579,736,444,  
228,676,544,198,707,211,501,466,617,215,819,586,483,617,813,579,153,75,939,528,112,994,903,367,182,413,367,444,156,869,886,716,707,660,495,196,736,199,337,292,142,22  
0,553,468,109,519,446,996,667,599,679,371,271,438,267,252,492,627,950,183,867,325,366,627,54,528,714,134,188,485,366,276,374,224,747,714,577,889,983,781,771,144,556,739,  
71,556,858,483,649,546,576,501,487,528,816,556,638,156,617,367,417,378,227,75,736,785,724,556,144,583,588,182,43,116,551,681,494,928,607,378,891,319,723,112,855,356,319,84,73  
5,374,241,199,492,581,581,336,492,104,989,167,173,436,211,208,619,366,198,152,846,848,726,228,629,266,610,944,776,18,411,785,421,118,492,110,757,638,199,883,598,130,5  
589,544,556,784,362,714,363,687,754,446,410,899,771,629,417,819,528,928,627,496,447,996,798,720,819,252,866,444,713,278,337,211,627,188,337,134,781,590,411,366,424,521,200,984,511,878,29,183,26,406,886,733,667,4  
79,899,501,615,485,976,845,945,615,813,417,229,252,843,53,156,891,574,576,747,747,339,211,627,188,337,134,781,590,411,366,424,521,200,984,511,878,29,183,26,406,886,733,667,4  
96,747,893,89,446,366,4,994,944,519,899,112,783,48,713,488,189,916,825,73,568,146,266,125,14,356,912,785,727,4,727,356,149,110,569,147,315,478,87,915,671,198,483,813,75,362  
,783,478,727,367,813,983,4,899,627,488,694,339,54,4,686,999,447,999,220,413,607,483,265,764,514,508,699,897,845,511,492,488,892,825,313,619,886,808,220,934,583,732,125,889,220,337  
862,771,266,466,945,147,568,337,519,920,511,228,589,974,864,372,512,224,436,686,983,617,752,151,881,998,224,68,161,781,882,276,882,446,784,553,556,336,988,336,556,15  
1 449 889 846 226 967 556 436 568 185 26 727 895 5 556 598 576 598 579 589 605 915 713 21 736 629 724 776 728 405 65 43 589 787 336 49 888 438 721 727 18 48 886 189 15 81
```

Рисунок 14 – Сгенерированная последовательность

Исходный код программы:

```
void rc4Func(string codeMethod, vector <int> genVec, int NumCount, const char* fileName,  
→ bool defaultVec) {  
    // Input: 256 initial values  
    if (defaultVec) {
```

```

genVec = { 802, 720, 341, 337, 961, 882, 417, 785, 198, 727, 899, 372, 374,
    ↪ 425, 556, 615, 813, 768, 840, 183, 893, 568, 73, 387, 18, 436, 182, 125,
    ↪ 806, 899, 485, 607, 619, 825, 944, 579, 707, 360, 363, 904, 87, 262,
    ↪ 276, 460, 687, 831, 75, 499, 599, 915, 681, 492, 483, 754, 878, 500,
    ↪ 189, 60, 624, 994, 959, 109, 600, 577, 934, 544, 156, 640, 903, 519,
    ↪ 544, 990, 781, 819, 449, 468, 650, 524, 967, 248, 438, 647, 739, 920,
    ↪ 400, 617, 419, 588, 676, 43, 581, 634, 151, 181, 211, 84, 724, 367, 723,
    ↪ 627, 886, 267, 617, 667, 85, 65, 134, 735, 589, 100, 983, 26, 747, 721,
    ↪ 945, 147, 337, 364, 734, 13, 406, 315, 647, 556, 496, 858, 640, 220,
    ↪ 224, 362, 847, 110, 629, 463, 776, 713, 528, 909, 448, 116, 9, 430, 141,
    ↪ 755, 151, 86, 901, 488, 449, 635, 500, 855, 950, 147, 410, 446, 4, 49,
    ↪ 665, 227, 411, 511, 336, 39, 974, 112, 752, 501, 21, 200, 617, 29, 629,
    ↪ 757, 784, 779, 843, 684, 266, 292, 319, 766, 146, 269, 912, 556, 714,
    ↪ 916, 605, 378, 142, 15, 889, 478, 54, 862, 590, 806, 363, 610, 5, 979,
    ↪ 638, 634, 736, 421, 413, 578, 105, 679, 869, 424, 444, 14, 692, 356,
    ↪ 569, 405, 271, 173, 783, 413, 188, 671, 891, 242, 533, 480, 48, 895, 89,
    ↪ 53, 873, 727, 686, 608, 147, 98, 185, 252, 776, 54, 675, 220, 67, 366,
    ↪ 576, 636, 771, 846, 808, 553, 259, 996, 224, 149 };
}

vector<int> k = genVec;
vector<int> s = genVec;
int j = 0;
for (int i = 0; i < 256; i++)
{
    j = (j + s[i] + k[i]) % 256;
    swap(s[i], s[j]);
}
int i = 0;
j = 0;

vector<int> showsVector = progressVector(NumCount);
int check_progress_index = 0;

ofstream f;
f.open(fileName, ios::out);
for (int countN = 0; countN < NumCount; countN++)
{
    i = (i + 1) % 256;
    j = (j + s[j]) % 256;
    swap(s[i], s[j]);
    int t = (s[i] + s[j]) % 256;
    if (defaultVec)
        f << s[t] % 1001 << ",";
    else
        f << s[t] << ",";
}

if (countN == showsVector[check_progress_index]) {
    cout << (check_progress_index + 1) * 10 << "% completed\n";
}

```

```

        check_progress_index++;
    }
}

f.close();
return;
}

```

1.8 ГПСЧ на основе RSA

Описание алгоритма:

1. Инициализация простых чисел p, q и $n = pq$, а также $f = (p - 1)(q - 1)$.
Выбрать случайное число $e: 1 < e < f$, $\text{НОД}(e, f) = 1$.
2. Выбрать случайное целое x_0 из интервала $[1, n - 1]$.
3. For $i = 1$ to w do
 - a) $x_i \leftarrow x_{i-1}^e \bmod n$.
 - б) $z_i \leftarrow$ последний значащий бит x_i
4. Вернуть z_1, \dots, z_w .

Инициализирующий вектор: модуль n , число e , начальное значение x .

Параметры запуска программы:

prng.exe /n:16000 /g:rsa /i:7191817,151,69 /f:prngTestRsa.txt

```

D:\cpp_projects\prng>prng.exe /n:16000 /g:rsa /i:7191817,151,69 /f:prngTestRsa.txt
The process of generating pseudo-random numbers:

10% completed
20% completed
30% completed
40% completed
50% completed
60% completed
70% completed
80% completed
90% completed
100% completed

Generation method: rsa
Count of generated numbers: 16000
Full name of the output file: prngTestRsa.txt

```

Рисунок 15 – Запуск программы

Рисунок 16 – Сгенерированная последовательность

Исходный код программы:

```
void rsaFunc(string codeMethod, vector <int> genVec, int NumCount, const char* fileName,
    bool defaultVec) {
    // Input: module n, number e, initial value of x; e satisfies conditions: 1 < e <
    // (p-1)(q-1), NOD(e, (p-1)(q-1)) = 1, where p*q=n. x from the interval [1,n]
    if (defaultVec) {
        genVec = { 7191817, 151, 69 };
    }
    int n = genVec[0];
    int e = genVec[1];
    int x = genVec[2];
    int l = 20;

    if (e <= 1 || x < 1 || x > n - 1) {
        cout << "\nError. The parameters do not meet the requirements.\n";
        return;
    }

vector<int> showsVector = progressVector(NumCount);
int check_progress_index = 0;

ofstream f;
f.open(fileName, ios::out);
for (int countN = 0; countN < NumCount; countN++)
{
    string zs = "";
    for (int i = 0; i < l; i++)

```

```

{
    x = powmod(x, e, n);
    char z = '0';
    if (x % 2 != 0)
        z = '1';
    zs.push_back(z);
}
if (defaultVec)
f << binToDec(zs) % 1001 << ",";
else
f << binToDec(zs) << ",";

if (countN == showsVector[check_progress_index]) {
    cout << (check_progress_index + 1) * 10 << "% completed\n";
    check_progress_index++;
}
f.close();
return;
}

```

1.9 Алгоритм Блюм-Блюма-Шуба

Описание алгоритма:

На входе: Длина l .

На выходе: Последовательность псевдослучайных бит z_1, z_2, \dots, z_l .

1. Сгенерировать два простых числа p и q , сравнимых с 3 по модулю 4. Произведение этих чисел, n , является целым числом Блюма. Выберем другое случайное целое число x , взаимно простое с n .
2. Вычислим $x_0 = x^2 \pmod{n}$, которое будет начальным вектором.
3. For $i = 1$ to w do
 - a) $x_i \leftarrow x_{i-1}^2 \pmod{n}$.
 - б) $z_i \leftarrow$ последний значащий бит x_i
4. Вернуть z_1, \dots, z_w .

Инициализирующий вектор: начальное значение x (взаимно простое с n).

Параметры запуска программы:

prng.exe /n:16000 /g:bbs /i:8627 /f:prngTestBbs.txt

```
D:\cpp_projects\prng>prng.exe /n:16000 /g:bbs /i:8627 /f:prngTestBbs.txt

The process of generating pseudo-random numbers:

10% completed
20% completed
30% completed
40% completed
50% completed
60% completed
70% completed
80% completed
90% completed
100% completed

Generation method: bbs
Count of generated numbers: 16000
Full name of the output file: prngTestBbs.txt
```

Рисунок 17 – Запуск программы

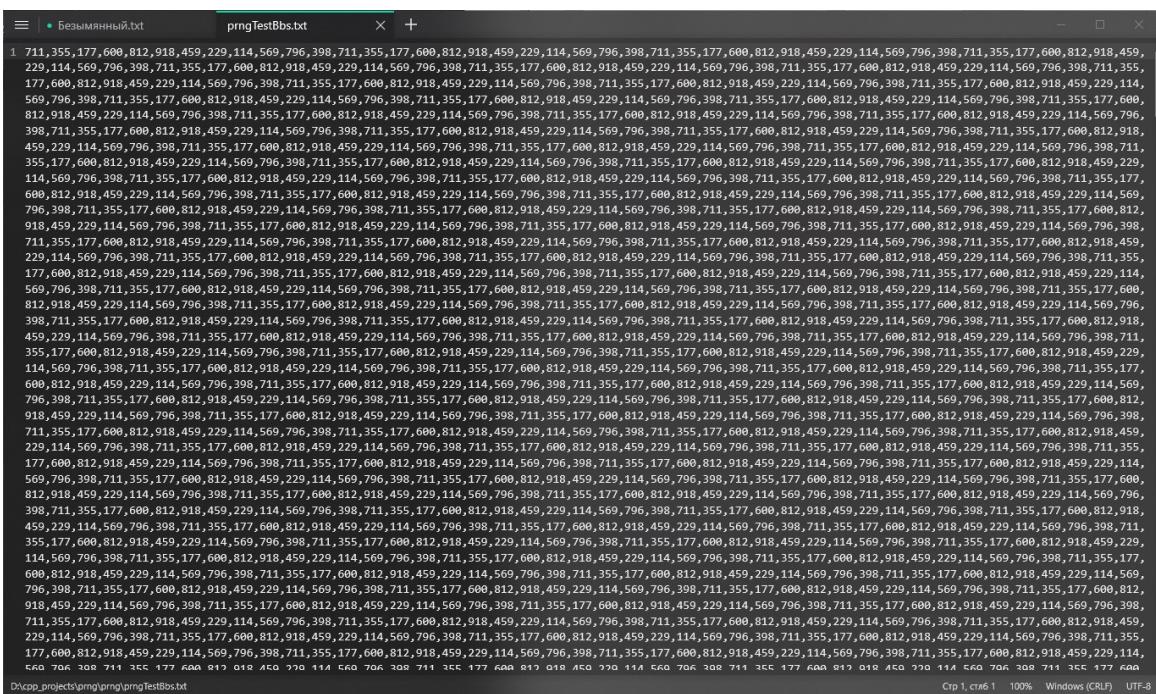


Рисунок 18 – Сгенерированная последовательность

Исходный код программы:

```
void bbsFunc(string codeMethod, vector <int> genVec, int NumCount, const char* fileName,
→ bool defaultVec) {
    // Input: Initial value of x (mutually prime with n)
    if (defaultVec) {
        genVec = { 8627 };
    }
    int x0 = genVec[0];
    int n = 16637;
    int l = 10;

    if (nod(x0, n) != 1) {
        cout << "\nThe parameters do not meet the requirements (x and n are not
→ mutually prime).\n";
    }
}
```

```

vector<int> showsVector = progressVector(NumCount);
int check_progress_index = 0;

ofstream f;
f.open(fileName, ios::out);
for (int countN = 0; countN < NumCount; countN++)
{
    x0 = (x0 * x0) % n;
    string zs = "";
    for (int i = 0; i < l; i++)
    {
        x0 = (x0 * x0) % n;
        char z = '0';
        if (x0 % 2 != 0)
            z = '1';
        zs.push_back(z);
    }
    if (defaultVec)
        f << binToDec(zs) % 1001 << ",";
    else
        f << binToDec(zs) << ",";
    if (countN == showsVector[check_progress_index]) {
        cout << (check_progress_index + 1) * 10 << "% completed\n";
        check_progress_index++;
    }
}
f.close();
return;
}

```

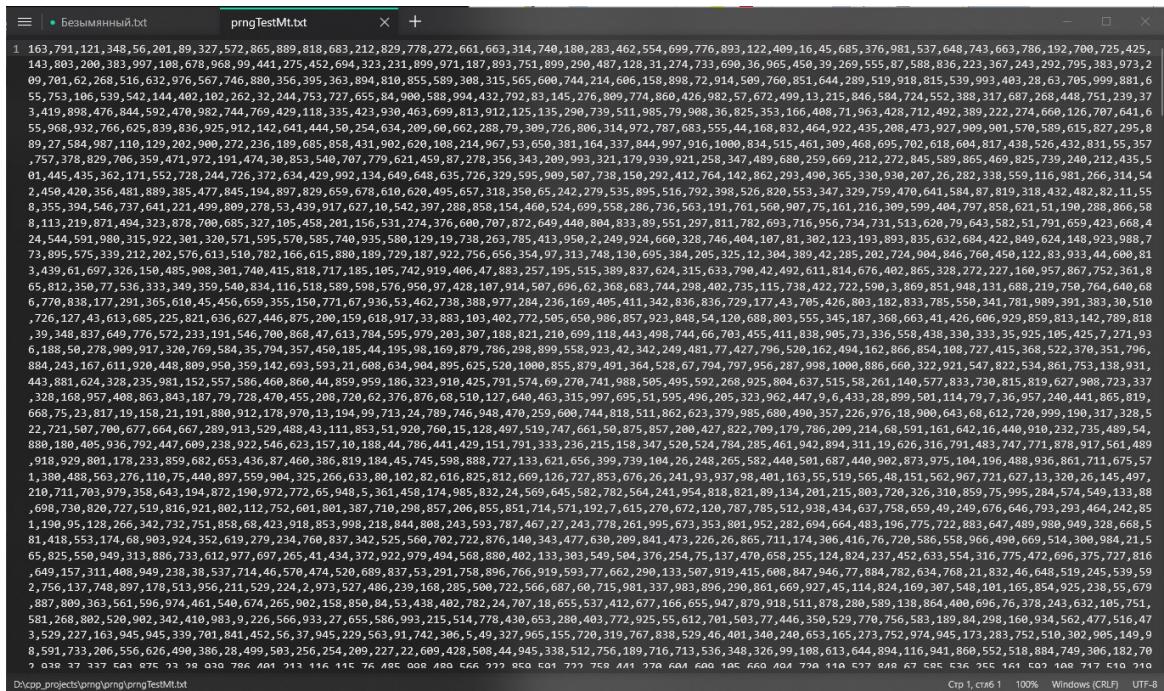
2 Преобразование ПСЧ к заданному распределению

Описание задания:

Создать программу для преобразования последовательности ПСЧ в другую последовательность ПСЧ с заданным распределением:

1. Стандартное равномерное с заданным интервалом;
2. Треугольное распределение;
3. Общее экспоненциальное распределение;
4. Нормальное распределение;
5. Гамма распределение (для параметра $c=k$);
6. Логнормальное распределение;
7. Логистическое распределение;
8. Биномиальное распределение.

Для преобразования последовательности ПСЧ в другую последовательность ПСЧ с заданным распределением использовалась последовательность, полученная с помощью вихря Мерсенна:



```
163,791,121,348,56,201,89,327,572,865,889,818,683,212,829,778,272,661,663,314,740,180,283,462,554,699,776,893,122,409,16,45,685,376,981,537,648,743,663,786,192,700,725,425,143,803,206,383,997,108,678,968,99,441,275,452,694,323,231,899,971,187,893,751,899,290,487,128,31,274,733,698,36,965,458,39,269,555,87,588,836,223,367,243,292,795,383,973,09,781,62,268,516,632,976,567,746,880,356,395,363,894,810,855,589,308,315,565,600,744,214,666,158,898,72,914,589,766,851,644,289,519,918,815,539,993,403,28,63,705,999,881,655,753,186,539,542,144,402,192,262,32,244,753,227,655,84,900,588,994,432,792,143,145,276,889,774,860,426,982,57,672,499,13,215,846,584,724,552,388,317,687,268,448,751,239,373,419,898,476,844,592,470,982,744,769,429,118,335,423,930,463,699,813,912,125,135,298,739,511,79,989,36,825,353,166,488,71,963,428,712,492,389,222,274,668,126,767,641,655,968,932,766,625,839,836,925,792,142,641,444,295,722,655,84,900,588,994,432,792,143,145,276,889,774,860,426,982,57,672,499,13,215,846,584,724,552,388,317,687,268,448,751,239,378,99,27,584,987,110,129,202,909,272,236,189,685,858,431,902,620,108,214,967,53,650,381,164,337,844,997,916,1000,834,515,461,309,468,695,702,618,604,817,438,526,432,831,55,357,757,378,82,766,859,471,972,191,474,38,853,540,707,779,621,459,87,278,356,343,269,993,321,179,939,921,258,347,489,688,259,669,212,272,845,589,865,469,825,739,248,212,435,501,445,435,362,192,152,278,244,276,372,634,429,992,134,649,648,635,726,329,595,989,587,738,158,292,412,764,142,862,293,496,365,338,938,287,26,282,338,559,116,981,266,314,542,458,420,356,481,889,385,477,845,194,889,829,659,678,610,628,495,657,318,358,65,242,279,535,895,516,792,398,526,828,553,347,329,759,470,641,584,87,819,318,432,482,711,558,8,355,394,546,737,641,221,499,889,278,53,439,917,627,10,542,397,288,553,454,656,254,699,558,286,736,563,191,761,569,987,75,161,216,428,621,51,190,288,466,588,113,219,871,494,323,878,700,685,327,105,458,261,156,531,274,376,600,707,872,649,448,804,833,89,551,297,811,782,693,716,956,734,731,513,620,79,643,582,51,791,659,423,668,8,24,544,591,980,315,922,301,320,571,595,570,585,740,935,588,129,19,738,263,785,413,950,2,249,924,660,726,306,314,972,787,683,555,44,169,832,464,922,435,208,473,927,909,901,570,589,615,827,295,889,27,584,987,110,129,202,909,272,236,189,685,858,431,902,620,108,214,967,53,650,381,164,337,844,997,916,1000,834,515,461,309,468,695,702,618,604,817,438,526,432,831,55,357,757,378,82,766,859,471,972,191,474,38,853,540,707,779,621,459,87,278,356,343,269,993,321,179,939,921,258,347,489,688,259,669,212,272,845,589,865,469,825,739,248,212,435,501,445,435,362,192,152,278,244,276,372,634,429,992,134,649,648,635,726,329,595,989,587,738,158,292,412,764,142,862,293,496,365,338,938,287,26,282,338,559,116,981,266,314,542,458,420,356,481,889,385,477,845,194,889,829,659,678,610,628,495,657,318,358,65,242,279,535,895,516,792,398,526,828,553,347,329,759,470,641,584,87,819,318,432,482,711,558,8,355,394,546,737,641,221,499,889,278,53,439,917,627,10,542,397,288,553,454,656,254,699,558,286,736,563,191,761,569,987,75,161,216,428,621,51,190,288,466,588,113,219,871,494,323,878,700,685,327,105,458,261,156,531,274,376,600,707,872,649,448,804,833,89,551,297,811,782,693,716,956,734,731,513,620,79,643,582,51,791,659,423,668,8,24,544,591,980,315,922,301,320,571,595,570,585,740,935,588,129,19,738,263,785,413,950,2,249,924,660,726,306,314,972,787,683,555,44,169,832,464,922,435,208,473,927,909,901,570,589,615,827,295,889,27,584,987,110,129,202,909,272,236,189,685,858,431,902,620,108,214,967,53,650,381,164,337,844,997,916,1000,834,515,461,309,468,695,702,618,604,817,438,526,432,831,55,357,757,378,82,766,859,471,972,191,474,38,853,540,707,779,621,459,87,278,356,343,269,993,321,179,939,921,258,347,489,688,259,669,212,272,845,589,865,469,825,739,248,212,435,501,445,435,362,192,152,278,244,276,372,634,429,992,134,649,648,635,726,329,595,989,587,738,158,292,412,764,142,862,293,496,365,338,938,287,26,282,338,559,116,981,266,314,542,458,420,356,481,889,385,477,845,194,889,829,659,678,610,628,495,657,318,358,65,242,279,535,895,516,792,398,526,828,553,347,329,759,470,641,584,87,819,318,432,482,711,558,8,355,394,546,737,641,221,499,889,278,53,439,917,627,10,542,397,288,553,454,656,254,699,558,286,736,563,191,761,569,987,75,161,216,428,621,51,190,288,466,588,113,219,871,494,323,878,700,685,327,105,458,261,156,531,274,376,600,707,872,649,448,804,833,89,551,297,811,782,693,716,956,734,731,513,620,79,643,582,51,791,659,423,668,8,24,544,591,980,315,922,301,320,571,595,570,585,740,935,588,129,19,738,263,785,413,950,2,249,924,660,726,306,314,972,787,683,555,44,169,832,464,922,435,208,473,927,909,901,570,589,615,827,295,889,27,584,987,110,129,202,909,272,236,189,685,858,431,902,620,108,214,967,53,650,381,164,337,844,997,916,1000,834,515,461,309,468,695,702,618,604,817,438,526,432,831,55,357,757,378,82,766,859,471,972,191,474,38,853,540,707,779,621,459,87,278,356,343,269,993,321,179,939,921,258,347,489,688,259,669,212,272,845,589,865,469,825,739,248,212,435,501,445,435,362,192,152,278,244,276,372,634,429,992,134,649,648,635,726,329,595,989,587,738,158,292,412,764,142,862,293,496,365,338,938,287,26,282,338,559,116,981,266,314,542,458,420,356,481,889,385,477,845,194,889,829,659,678,610,628,495,657,318,358,65,242,279,535,895,516,792,398,526,828,553,347,329,759,470,641,584,87,819,318,432,482,711,558,8,355,394,546,737,641,221,499,889,278,53,439,917,627,10,542,397,288,553,454,656,254,699,558,286,736,563,191,761,569,987,75,161,216,428,621,51,190,288,466,588,113,219,871,494,323,878,700,685,327,105,458,261,156,531,274,376,600,707,872,649,448,804,833,89,551,297,811,782,693,716,956,734,731,513,620,79,643,582,51,791,659,423,668,8,24,544,591,980,315,922,301,320,571,595,570,585,740,935,588,129,19,738,263,785,413,950,2,249,924,660,726,306,314,972,787,683,555,44,169,832,464,922,435,208,473,927,909,901,570,589,615,827,295,889,27,584,987,110,129,202,909,272,236,189,685,858,431,902,620,108,214,967,53,650,381,164,337,844,997,916,1000,834,515,461,309,468,695,702,618,604,817,438,526,432,831,55,357,757,378,82,766,859,471,972,191,474,38,853,540,707,779,621,459,87,278,356,343,269,993,321,179,939,921,258,347,489,688,259,669,212,272,845,589,865,469,825,739,248,212,435,501,445,435,362,192,152,278,244,276,372,634,429,992,134,649,648,635,726,329,595,989,587,738,158,292,412,764,142,862,293,496,365,338,938,287,26,282,338,559,116,981,266,314,542,458,420,356,481,889,385,477,845,194,889,829,659,678,610,628,495,657,318,358,65,242,279,535,895,516,792,398,526,828,553,347,329,759,470,641,584,87,819,318,432,482,711,558,8,355,394,546,737,641,221,499,889,278,53,439,917,627,10,542,397,288,553,454,656,254,699,558,286,736,563,191,761,569,987,75,161,216,428,621,51,190,288,466,588,113,219,871,494,323,878,700,685,327,105,458,261,156,531,274,376,600,707,872,649,448,804,833,89,551,297,811,782,693,716,956,734,731,513,620,79,643,582,51,791,659,423,668,8,24,544,591,980,315,922,301,320,571,595,570,585,740,935,588,129,19,738,263,785,413,950,2,249,924,660,726,306,314,972,787,683,555,44,169,832,464,922,435,208,473,927,909,901,570,589,615,827,295,889,27,584,987,110,129,202,909,272,236,189,685,858,431,902,620,108,214,967,53,650,381,164,337,844,997,916,1000,834,515,461,309,468,695,702,618,604,817,438,526,432,831,55,357,757,378,82,766,859,471,972,191,474,38,853,540,707,779,621,459,87,278,356,343,269,993,321,179,939,921,258,347,489,688,259,669,212,272,845,589,865,469,825,739,248,212,435,501,445,435,362,192,152,278,244,276,372,634,429,992,134,649,648,635,726,329,595,989,587,738,158,292,412,764,142,862,293,496,365,338,938,287,26,282,338,559,116,981,266,314,542,458,420,356,481,889,385,477,845,194,889,829,659,678,610,628,495,657,318,358,65,242,279,535,895,516,792,398,526,828,553,347,329,759,470,641,584,87,819,318,432,482,711,558,8,355,394,546,737,641,221,499,889,278,53,439,917,627,10,542,397,288,553,454,656,254,699,558,286,736,563,191,761,569,987,75,161,216,428,621,51,190,288,466,588,113,219,871,494,323,878,700,685,327,105,458,261,156,531,274,376,600,707,872,649,448,804,833,89,551,297,811,782,693,716,956,734,731,513,620,79,643,582,51,791,659,423,668,8,24,544,591,980,315,922,301,320,571,595,570,585,740,935,588,129,19,738,263,785,413,950,2,249,924,660,726,306,314,972,787,683,555,44,169,832,464,922,435,208,473,927,909,901,570,589,615,827,295,889,27,584,987,110,129,202,909,272,236,189,685,858,431,902,620,108,214,967,53,650,381,164,337,844,997,916,1000,834,515,461,309,468,695,702,618,604,817,438,526,432,831,55,357,757,378,82,766,859,471,972,191,474,38,853,540,707,779,621,459,87,278,356,343,269,993,321,179,939,921,258,347,489,688,259,669,212,272,845,589,865,469,825,739,248,212,435,501,445,435,362,192,152,278,244,276,372,634,429,992,134,649,648,635,726,329,595,989,587,738,158,292,412,764,142,862,293,496,365,338,938,287,26,282,338,559,116,981,266,314,542,458,420,356,481,889,385,477,845,194,889,829,659,678,610,628,495,657,318,358,65,242,279,535,895,516,792,398,526,828,553,347,329,759,470,641,584,87,819,318,432,482,711,558,8,355,394,546,737,641,221,499,889,278,53,439,917,627,10,542,397,288,553,454,656,254,699,558,286,736,563,191,761,569,987,75,161,216,428,621,51,190,288,466,588,113,219,871,494,323,878,700,685,327,105,458,261,156,531,274,376,600,707,872,649,448,804,833,89,551,297,811,782,693,716,956,734,731,513,620,79,643,582,51,791,659,423,668,8,24,544,591,980,315,922,301,320,571,595,570,585,740,935,588,129,19,738,263,785,413,950,2,249,924,660,726,306,314,972,787,683,555,44,169,832,464,922,435,208,473,927,909,901,570,589,615,827,295,889,27,584,987,110,129,202,909,272,236,189,685,858,431,902,620,108,214,967,53,650,381,164,337,844,997,916,1000,834,515,461,309,468,695,702,618,604,817,438,526,432,831,55,357,757,378,82,766,859,471,972,191,474,38,853,540,707,779,621,459,87,278,356,343,269,993,321,179,939,921,258,347,489,688,259,669,212,272,845,589,865,469,825,739,248,212,435,501,445,435,362,192,152,278,244,276,372,634,429,992,134,649,648,635,726,329,595,989,587,738,158,292,412,764,142,862,293,496,365,338,938,287,26,282,338,559,116,981,266,314,542,458,420,356,481,889,385,477,845,194,889,829,659,678,610,628,495,657,318,358,65,242,279,535,895,516,792,398,526,828,553,347,329,759,470,641,584,87,819,318,432,482,711,558,8,355,394,546,737,641,221,499,889,278,53,439,917,627,10,542,397,288,553,454,656,254,699,558,286,736,563,191,761,569,987,75,161,216,428,621,51,190,288,466,588,113,219,871,494,323,878,700,685,327,105,458,261,156,531,274,376,600,707,872,649,448,804,833,89,551,297,811,782,693,716,956,734,731,513,620,79,643,582,51,791,659,423,668,8,24,544,591,980,315,922,301,320,571,595,570,585,740,935,588,129,19,738,263,785,413,950,2,249,924,660,726,306,314,972,787,683,555,44,169,832,464,922,435,208,473,927,909,901,
```

2.1 Стандартное равномерное с заданным интервалом

Описание алгоритма:

Если число U соответствует стандартному равномерному распределению, то равномерное случайное число должно быть получено в соответствии со следующей формулой:

$$Y = bU + a$$

Параметры запуска программы:

rnc.exe /d:st /f:prngTestMt.txt /p1:75 /p2:200

```
D:\cpp_projects\rnc\rnc>rnc.exe /d:st /f:prngTestMt.txt /p1:75 /p2:200

Distribution method: st
Parameter 1: 75
Parameter 2: 200
Input File: prngTestMt.txt
Output File: stOutput.dat
```

Рисунок 20 – Запуск программы

Рисунок 21 – Последовательность с заданным распределением

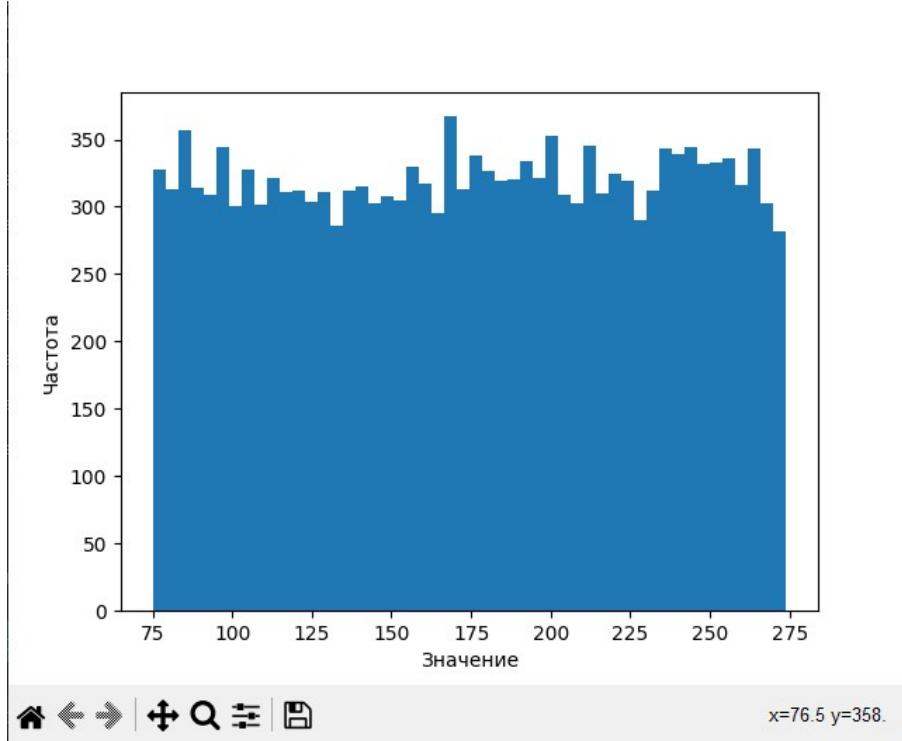


Рисунок 22 – График распределения

Исходный код программы:

```

vector <double> stFunc(string distributionMethod, string inputFileName, string parameter1,
→  string parameter2, bool defaultVec, bool& errorMessage) {
    ifstream inFile(inputFileName);
    string lineFile;
    vector <double> allElems;
    if (!inFile.is_open()) {
        cout << "Ошибка открытия файла " << inputFileName << endl;
        errorMessage = true;
        return allElems;
    }
    string getlineElem;
    getline(inFile, getlineElem);
    stringstream ss(getlineElem);
    while (getline(ss, getlineElem, ',')) {
        double intElem = stod(getlineElem);
        allElems.push_back(intElem);
    }
    inFile.close();

    vector <double> allU = XtoU(allElems);
    double a = stod(parameter1);
    double b = stod(parameter2);
    for (int i = 0; i < allU.size(); i++)
    {
        allU[i] = (b * allU[i]) + a;
    }
}

```

```

    }

    return allU;
}

```

2.2 Треугольное распределение

Описание алгоритма:

Если стандартные случайные числа U_1 и U_2 независимо получены методом генерации стандартного равномерного числа, то случайное число Y , подчиняющееся треугольному распределению, определяют по формуле:

$$Y = a + b(U_1 + U_2 - 1)$$

Параметры запуска программы:

rnc.exe /d:tr /f:prngTestMt.txt /p1:1 /p2:1000

```
D:\cpp_projects\rnc\rnc>rnc.exe /d:tr /f:prngTestMt.txt /p1:1 /p2:1000
Distribution method: tr
Parameter 1: 1
Parameter 2: 1000
Input File: prngTestMt.txt
Output File: trOutput.dat
```

Рисунок 23 – Запуск программы

```
1,-45,-87,-530,-595,-742,-789,-583,-100,436,753,796,500,-104,406,606,49,-66,323,-22,53,-79,-536,-254,15,252,474,668,14,-468,-574,-938,-269,60,356,517,184,390,405,448,-21,-187,424,149,-431,-53,-2,-416,379,184,-213,645,66,-459,-283,-272,145,16,-445,129,869,157,79,643,649,188,-222,-384,-840,-694,6,422,-273,1,414,-510,-691,-175,-357,-324,423,58,-409,-389,-464,86,177,355,181,-89,-236,-669,-215,147,607,542,312,625,235,-248,-241,256,783,664,443,-102,-376,-119,164,343,-41,-179,-235,55,-29,-13,422,268,618,494,-66,-191,436,732,353,531,395,-568,-908,-231,783,879,535,407,-146,-354,88,-313,-453,-495,-693,-785,-23,-247,93,381,-260,-15,487,581,425,223,-124,-771,-578,84,582,633,285,497,38,-270,170,-487,-771,60,429,307,275,-59,-294,3,-44,-283,-198,-9,-387,-207,-316,373,319,435,61,451,725,512,197,-452,-546,-241,352,392,161,511,724,36,-739,-574,-28,249,495,63,-12,-55,-138,177,-488,-425,-520,33,390,139,283,-118,-388,-503,-65,-213,-166,347,295,622,899,697,394,463,674,768,836,53,-216,84,-505,-695,-111,-156,-730,-277,-49,-632,-611,34,531,119,285,-788,469,237,-400,-787,0,295,385,356,-356,-318,399,835,889,470,158,283,441,121,183,-83,-388,578,96,-768,-668,191,171,-491,-574,-125,542,282,332,523,-271,-677,188,19,-296,38,-454,-498,188,840,912,915,833,348,-23,-261,-584,-692,-478,-758,235,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419,-143,-7,-128,-223,-162,369,273,-137,321,38,99,725,487,336,287,229,114,151,-24,-331,-584,-692,-478,-185,429,410,387,189,-75,345,372,-99,-233,87,228,110,224,-328,-93,136,-245,-38,-495,-116,392,246,485,399,79,-453,-634,-365,-300,-447,281,313,-491,117,859,178,-399,-163,168,-60,-71,-118,-515,116,433,453,333,293,563,-20,-547,-352,-63,-53,-119,-202,-466,-276,279,-27,-29,97,5,62,420,125,-216,296,-282,369,54,-75,503,415,244,-111,-557,-295,175,-93,3,154,-216,-144,-304,259,136,-766,-691,-379,-102,-324,96,246,-419
```

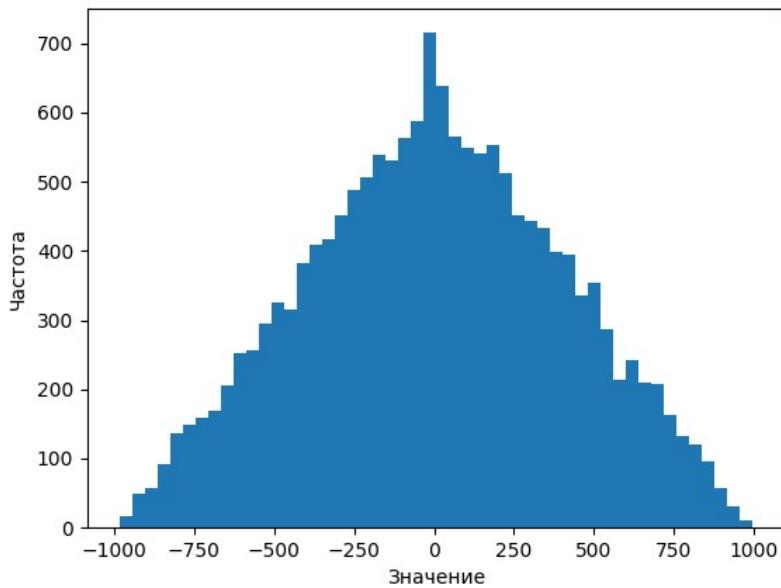


Рисунок 25 – График распределения

Исходный код программы:

```

vector <double> trFunc(string distributionMethod, string inputFileName, string parameter1,
→  string parameter2, bool defaultVec, bool& errorMessage) {
    ifstream inFile(inputFileName);
    string lineFile;
    vector <double> allElems;
    if (!inFile.is_open()) {
        cout << "Ошибка открытия файла " << inputFileName << endl;
        errorMessage = true;
        return allElems;
    }
    string getlineElem;
    getline(inFile, getlineElem);
    stringstream ss(getlineElem);
    while (getline(ss, getlineElem, ',')) {
        double intElem = stod(getlineElem);
        allElems.push_back(intElem);
    }
    inFile.close();

    vector <double> allU = XtoU(allElems);
    double a = stod(parameter1);
    double b = stod(parameter2);
    for (int i = 0; i < allU.size() - 1; i++)
    {
        allU[i] = a + (b * (allU[i] + allU[i + 1] - 1));
    }
}

```

```
    }  
  
    return allU;  
}
```

2.3 Общее экспоненциальное распределение

Описание алгоритма:

Если стандартное равномерное случайное число U генерировано одним из методов, установленным в разделе 2, то случайное число, соответствующее экспоненциальному распределению, получают по формуле:

$$Y = -b \ln(U) + a$$

Параметры запуска программы:

rnc.exe /d:ex /f:D:prngTestMt.txt /p1:7 /p2:12

```
D:\cpp_projects\rnc\rnc>rnc.exe /d:ex /f:D:\prngTestMt.txt /p1:7 /p2:12
Distribution method: ex
Parameter 1: 7
Parameter 2: 12
Input File: D:\prngTestMt.txt
Output File: exOutput.dat
```

Рисунок 26 – Запуск программы

Рисунок 27 – Последовательность с заданным распределением

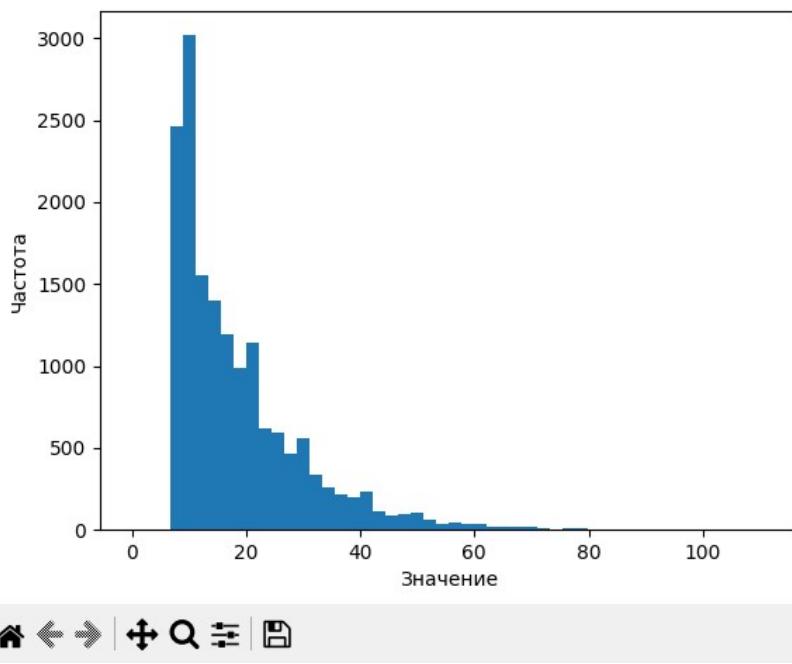


Рисунок 28 – График распределения

Исходный код программы:

```

vector <double> exFunc(string distributionMethod, string inputFileName, string parameter1,
→   string parameter2, bool defaultVec, bool& errorMessage) {
    ifstream inFile(inputFileName);
    string lineFile;
    vector <double> allElems;
    if (!inFile.is_open()) {
        cout << "Ошибка открытия файла " << inputFileName << endl;
        errorMessage = true;
        return allElems;
    }
    string getlineElem;
    getline(inFile, getlineElem);
    stringstream ss(getlineElem);
    while (getline(ss, getlineElem, ',')) {
        double intElem = stod(getlineElem);
        allElems.push_back(intElem);
    }
    inFile.close();

    vector <double> allU = XtoU(allElems);
    double a = stod(parameter1);
    double b = stod(parameter2);
    for (int i = 0; i < allU.size() - 1; i++) {
        if (allU[i] == 0) {
    
```

```

        cout << "Одно из случайных чисел U = 0. Ошибка в вычислении ln(U)"
        << endl;
    errorMessage = true;
    return allU;
}
allU[i] = ((b * (-1)) * log(allU[i])) + a;
}

return allU;
}

```

2.4 Нормальное распределение

Описание алгоритма:

Если стандартные равномерные случайные числа U_1 и U_2 независимо сгенерированы методом, установленным в разделе 2, то два независимых нормальных случайных числа Z_1, Z_2 получают в соответствии со следующей процедурой

$$Z_1 = \mu + \sigma \sqrt{-2 \ln(1 - U_1)} \cos(2\pi U_2)$$

$$Z_2 = \mu + \sigma \sqrt{-2 \ln(1 - U_1)} \sin(2\pi U_2)$$

Параметры запуска программы:

rnc.exe /d:nr /f:D:prngTestMt.txt /p1:1 /p2:1000

```
D:\cpp_projects\rnc>rnc.exe /d:nr /f:D:prngTestMt.txt /p1:1 /p2:1000
Distribution method: nr
Parameter 1: 1
Parameter 2: 1000
Input File: D:prngTestMt.txt
Output File: nrOutput.dat
```

Рисунок 29 – Запуск программы

Рисунок 30 – Последовательность с заданным распределением

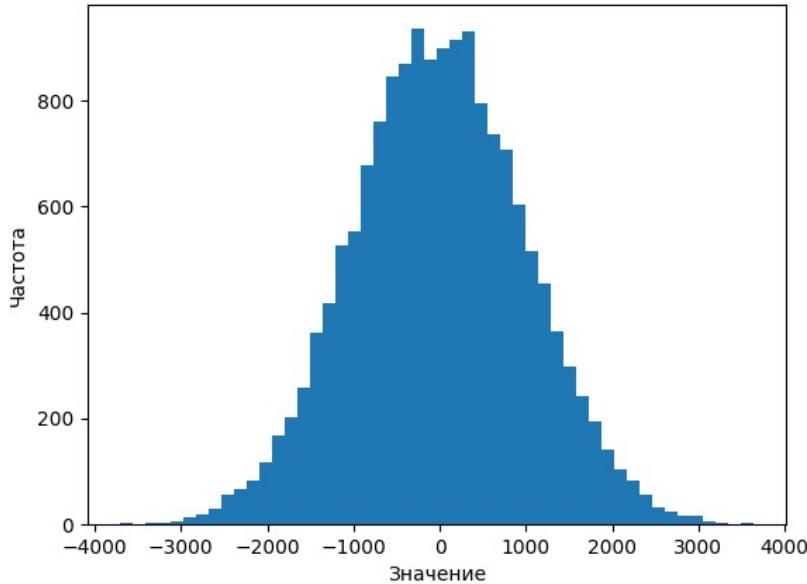


Рисунок 31 – График распределения

Исходный код программы:

```
vector <double> nrFunc(string distributionMethod, string inputFileName, string parameter1,  
→   string parameter2, bool defaultVec, bool& errorMessage) {  
    ifstream inFile(inputFileName);  
    string lineFile;
```

```

vector <double> allElems;
if (!inFile.is_open()) {
    cout << "Ошибка открытия файла " << inputFileName << endl;
    errorMessage = true;
    return allElems;
}
string getlineElem;
getline(inFile, getlineElem);
stringstream ss(getlineElem);
while (getline(ss, getlineElem, ',')) {
    double intElem = stod(getlineElem);
    allElems.push_back(intElem);
}
inFile.close();

if (allElems.size() % 2 != 0)
allElems.erase(allElems.end() - 1);

vector <double> allU = XtoU(allElems);
double mu = stod(parameter1);
double sigma = stod(parameter2);
for (int i = 0; i < allU.size(); i += 2)
{
    if (allU[i] == 1) {
        cout << "Одно из случайных чисел U = 1. Ошибка в вычислении ln(1 -
        ↪ U)" << endl;
        errorMessage = true;
        return allU;
    }
    double U1 = allU[i];
    double U2 = allU[i + 1];
    allU[i] = mu + (sigma * sqrt(-2 * log(1 - U1)) * cos(2 * M_PI * U2)); // Z1
    allU[i + 1] = mu + (sigma * sqrt(-2 * log(1 - U1)) * sin(2 * M_PI * U2)); // 
        ↪ Z2
}
return allU;
}

```

2.5 Гамма распределение

Описание алгоритма:

Используя независимую равномерную случайную величину U_1 , т.к. рассматривается случай ($c = 1$), применяем формулу

$$Y = a - b \ln(1 - U_1)$$

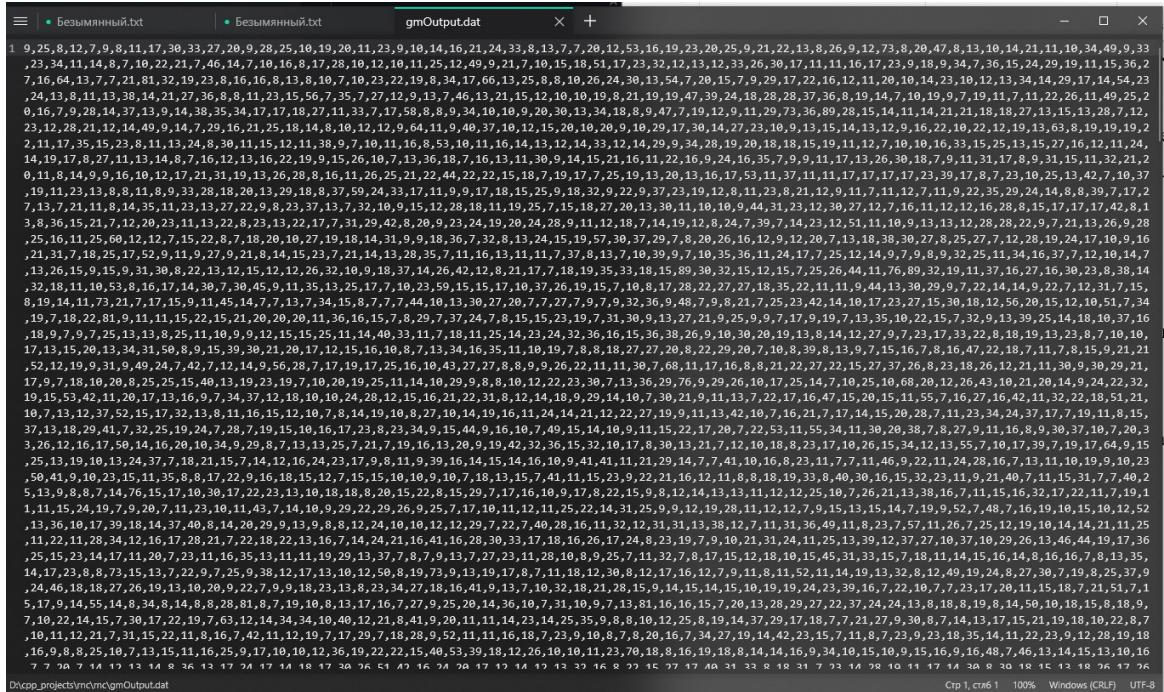
Параметры запуска программы:

rnc.exe /d:gm /f:D:prngTestMt.txt /p1:7 /p2:12

```
D:\cpp_projects\rnc\rnc>rnc.exe /d:gm /f:D:prngTestMt.txt /p1:7 /p2:12

Distribution method: gm
Parameter 1: 7
Parameter 2: 12
Input File: D:prngTestMt.txt
Output File: gmOutput.dat
```

Рисунок 32 – Запуск программы



```
1 9,25,8,12,7,9,8,11,17,38,33,27,20,9,28,25,18,9,20,11,23,9,18,14,16,21,24,33,8,13,7,28,12,53,16,19,23,20,25,9,21,22,13,8,26,9,12,73,8,20,47,8,13,10,14,21,11,18,34,49,9,33
,23,34,11,14,8,7,10,22,21,7,46,14,7,10,16,8,17,28,10,12,11,18,9,21,7,13,17,23,32,12,13,12,33,26,30,17,11,11,16,17,23,9,18,9,34,7,36,15,24,29,19,11,15,36,2
7,16,64,13,7,7,21,81,32,19,23,8,16,8,13,8,8,10,7,16,23,22,19,8,34,17,66,13,25,8,8,10,26,24,30,13,54,7,20,15,7,9,29,17,22,16,12,11,26,10,14,23,10,12,13,34,14,29,17,14,54,23
,24,13,8,11,13,38,14,21,27,36,8,8,11,23,15,56,7,35,7,27,12,9,13,7,46,13,21,15,12,10,10,8,21,19,47,39,24,18,28,37,36,8,19,14,7,10,19,9,7,19,11,7,11,22,26,11,49,25,2
0,16,7,9,28,14,37,13,9,14,38,35,34,17,17,18,27,11,33,7,17,58,8,8,9,34,18,9,20,30,13,34,18,8,9,47,7,19,12,9,11,29,73,36,89,28,15,14,11,14,21,21,18,27,13,15,13,28,7,12,
23,12,28,21,12,14,49,9,14,7,29,16,21,25,18,14,8,18,12,12,9,64,11,9,48,37,18,12,15,20,10,20,9,18,29,17,30,14,27,23,18,9,13,15,14,13,12,9,16,22,18,10,22,12,19,13,63,8,19,19,19,19,2
2,11,17,35,15,23,8,11,13,24,8,30,11,15,12,11,16,8,53,10,11,16,14,13,12,14,33,12,14,29,9,34,28,19,28,18,15,19,11,12,7,18,10,16,33,15,15,13,15,27,16,12,11,24,
14,19,17,8,27,11,13,14,8,7,16,12,13,16,22,19,9,15,26,19,17,3,36,18,7,16,13,11,30,9,14,15,21,16,11,22,16,9,24,16,35,7,9,9,11,17,13,26,38,18,7,9,11,31,17,8,9,31,15,11,32,21,2
0,11,8,14,9,9,16,10,12,17,21,31,19,13,26,28,8,16,11,26,25,21,22,15,18,7,19,17,7,25,19,13,20,13,16,17,53,11,37,11,11,17,17,17,17,23,39,17,8,7,23,10,25,13,42,7,10,37
,19,11,23,13,8,8,11,8,9,33,28,18,20,13,29,18,8,37,59,24,33,17,11,9,9,17,18,15,25,9,18,32,9,23,19,12,8,11,23,8,21,12,9,11,7,11,12,7,11,22,35,29,24,14,8,8,39,7,17,2
7,13,7,21,11,8,14,35,11,23,13,27,22,9,8,23,37,13,7,32,16,9,15,12,28,18,11,19,25,7,15,18,27,28,13,38,11,16,9,44,31,23,32,38,27,12,7,16,28,8,15,17,17,17,42,8,1
3,8,36,15,21,7,12,28,23,11,13,22,8,23,13,22,7,31,29,42,8,28,9,23,24,19,20,24,28,9,11,12,18,7,14,19,12,8,24,7,9,7,14,23,12,51,11,18,9,13,12,28,28,22,9,7,21,13,6,29,8
,25,16,11,25,68,12,12,7,15,22,8,7,15,20,19,11,14,31,9,9,18,36,7,32,8,13,24,15,19,57,30,37,29,7,8,20,26,16,12,9,12,20,7,13,18,38,30,27,8,25,27,7,12,28,19,24,17,18,9,16
,21,31,7,18,25,17,52,9,11,9,27,9,21,8,14,15,23,7,21,14,13,28,35,7,11,16,13,11,11,7,37,8,13,7,18,39,9,7,10,35,36,11,24,17,7,25,12,14,9,7,9,8,9,32,25,11,34,16,37,7,12,18,14,7
,13,26,15,9,15,9,31,38,8,22,13,12,15,12,26,32,10,9,18,37,14,26,42,12,15,12,15,7,25,26,44,11,76,89,32,19,11,37,16,27,16,30,38,8,38,14
,32,18,11,18,53,8,16,17,44,30,7,30,45,9,11,35,13,25,17,7,10,23,25,15,17,18,37,26,19,25,11,11,9,44,13,38,29,9,7,22,14,14,9,22,7,12,31,7,15,
8,19,14,11,73,21,7,17,15,9,11,45,14,7,7,13,7,34,15,8,7,7,7,44,10,13,30,27,28,7,7,27,7,9,7,9,32,36,9,48,7,9,8,21,7,25,23,42,14,10,17,23,27,15,30,18,12,56,20,15,12,19,51,7,34
,19,7,18,22,81,9,11,11,15,22,15,21,20,20,28,11,36,16,15,7,8,29,7,37,24,7,8,15,23,19,7,31,30,9,13,27,21,9,25,9,9,17,17,9,19,7,13,35,16,22,15,7,32,9,13,39,25,14,18,10,37,16
,18,9,7,9,25,13,13,8,25,11,9,15,15,25,11,14,40,33,11,15,18,11,25,14,23,24,32,36,16,15,36,38,26,9,10,30,28,19,13,8,14,12,27,9,7,23,17,33,22,8,18,19,13,23,8,7,10,18
,17,13,15,20,13,34,31,50,8,9,15,39,30,21,28,17,12,15,16,10,9,18,7,13,34,16,35,11,10,19,7,8,8,18,27,27,20,8,22,29,20,7,18,8,39,8,13,9,7,15,16,7,8,16,47,22,18,7,11,7,8,15,9,21,21
,52,12,19,9,31,9,49,24,7,42,12,14,9,56,28,7,17,19,17,23,14,12,15,12,15,12,15,7,25,26,44,11,76,89,32,19,11,37,16,27,16,30,38,8,38,14
,17,9,7,18,10,20,8,25,15,25,15,48,13,19,23,19,7,18,20,9,12,22,15,9,8,8,10,12,22,30,7,13,36,29,9,29,26,10,17,25,14,7,18,25,10,18,20,12,26,43,10,21,20,14,9,4,22,22,32
,19,15,53,42,11,20,17,13,16,9,7,34,37,12,18,10,24,28,12,15,21,22,31,9,12,14,18,9,29,14,10,7,38,21,9,11,13,7,22,17,16,47,15,20,20,15,11,55,7,16,27,16,42,11,32,22,18,51,21
,10,7,13,12,37,52,15,17,32,13,8,11,16,15,12,10,7,8,14,19,10,8,27,18,14,19,11,12,24,14,21,12,22,27,19,9,11,13,42,10,7,16,21,7,17,14,15,20,28,7,11,23,34,24,37,17,7,19,11,8,15
,37,13,18,29,41,7,32,25,19,24,7,28,7,19,15,16,16,17,23,8,23,34,9,15,44,9,18,10,7,49,15,14,18,9,11,15,22,17,28,7,22,53,11,55,34,11,30,28,38,7,8,27,9,11,16,8,9,38,37,18,7,20,3
,3,26,12,16,17,58,14,16,28,10,34,9,29,8,7,13,13,25,7,21,7,19,16,13,28,9,19,42,52,36,15,32,18,17,8,10,18,8,23,17,18,26,15,34,12,13,55,7,16,17,39,7,19,17,64,9,15
,25,13,19,18,13,37,7,18,21,15,7,14,12,16,24,23,17,9,8,11,9,39,16,14,16,35,11,10,19,7,8,8,18,27,27,20,8,22,29,20,7,18,8,39,8,13,9,7,15,16,7,8,16,47,22,18,7,11,7,8,15,9,21,21
,58,41,9,19,23,15,11,35,8,8,17,22,9,16,18,15,12,7,15,15,9,26,22,11,38,7,68,11,17,16,8,18,21,22,27,22,15,7,27,36,8,23,18,26,12,21,11,38,9,28,29,21
,17,9,7,18,10,20,8,25,15,25,15,48,13,19,23,19,7,18,20,9,12,22,30,7,13,36,29,9,29,26,10,17,25,14,7,18,25,10,18,20,12,26,43,10,21,20,14,9,4,22,22,32
,19,15,53,42,11,20,17,13,16,9,7,34,37,12,18,10,24,28,12,15,21,22,31,9,12,14,18,9,29,14,10,7,38,21,9,11,13,7,22,17,16,47,15,20,20,15,11,55,7,16,27,16,42,11,32,22,18,51,21
,10,7,13,12,37,52,15,17,32,13,8,11,16,15,12,10,7,8,14,19,10,8,27,18,14,19,11,12,24,14,21,12,22,27,19,9,11,13,42,10,7,16,21,7,17,14,15,20,28,7,11,23,34,24,37,17,7,19,11,8,15
,37,13,18,29,41,7,32,25,19,24,7,28,7,19,15,16,16,17,23,8,23,34,9,15,44,9,18,10,7,49,15,14,18,9,11,15,22,17,28,7,22,53,11,55,34,11,30,28,38,7,8,27,9,11,16,8,9,38,37,18,7,20,3
,3,26,12,16,17,58,14,16,28,10,34,9,29,8,7,13,13,25,7,21,7,19,16,13,28,9,19,42,52,36,15,32,18,17,8,10,18,8,23,17,18,26,15,34,12,13,55,7,16,17,39,7,19,17,64,9,15
,25,13,19,18,13,37,7,18,21,15,7,14,12,16,24,23,17,9,8,11,9,39,16,14,16,35,11,10,19,7,8,8,18,27,27,20,8,22,29,20,7,18,8,39,8,13,9,7,15,16,7,8,16,47,22,18,7,11,7,8,15,9,21,21
,58,41,9,19,23,15,11,35,8,8,17,22,9,16,18,15,12,7,15,15,9,26,22,11,38,7,68,11,17,16,8,18,21,22,27,22,15,7,27,36,8,23,18,26,12,21,11,38,9,28,29,21
,17,9,7,18,10,20,8,25,15,25,15,48,13,19,23,19,7,18,20,9,12,22,30,7,13,36,29,9,29,26,10,17,25,14,7,18,25,10,18,20,12,26,43,10,21,20,14,9,4,22,22,32
,19,15,53,42,11,20,17,13,16,9,7,34,37,12,18,10,24,28,12,15,21,22,31,9,12,14,18,9,29,14,10,7,38,21,9,11,13,7,22,17,16,47,15,20,20,15,11,55,7,16,27,16,42,11,32,22,18,51,21
,10,7,13,12,37,52,15,17,32,13,8,11,16,15,12,10,7,8,14,19,10,8,27,18,14,19,11,12,24,14,21,12,22,27,19,9,11,13,42,10,7,16,21,7,17,14,15,20,28,7,11,23,34,24,37,17,7,19,11,8,15
,37,13,18,29,41,7,32,25,19,24,7,28,7,19,15,16,16,17,23,8,23,34,9,15,44,9,18,10,7,49,15,14,18,9,11,15,22,17,28,7,22,53,11,55,34,11,30,28,38,7,8,27,9,11,16,8,9,38,37,18,7,20,3
,3,26,12,16,17,58,14,16,28,10,34,9,29,8,7,13,13,25,7,21,7,19,16,13,28,9,19,42,52,36,15,32,18,17,8,10,18,8,23,17,18,26,15,34,12,13,55,7,16,17,39,7,19,17,64,9,15
,25,13,19,18,13,37,7,18,21,15,7,14,12,16,24,23,17,9,8,11,9,39,16,14,16,35,11,10,19,7,8,8,18,27,27,20,8,22,29,20,7,18,8,39,8,13,9,7,15,16,7,8,16,47,22,18,7,11,7,8,15,9,21,21
,58,41,9,19,23,15,11,35,8,8,17,22,9,16,18,15,12,7,15,15,9,26,22,11,38,7,68,11,17,16,8,18,21,22,27,22,15,7,27,36,8,23,18,26,12,21,11,38,9,28,29,21
,17,9,7,18,10,20,8,25,15,25,15,48,13,19,23,19,7,18,20,9,12,22,30,7,13,36,29,9,29,26,10,17,25,14,7,18,25,10,18,20,12,26,43,10,21,20,14,9,4,22,22,32
,19,15,53,42,11,20,17,13,16,9,7,34,37,12,18,10,24,28,12,15,21,22,31,9,12,14,18,9,29,14,10,7,38,21,9,11,13,7,22,17,16,47,15,20,20,15,11,55,7,16,27,16,42,11,32,22,18,51,21
,10,7,13,12,37,52,15,17,32,13,8,11,16,15,12,10,7,8,14,19,10,8,27,18,14,19,11,12,24,14,21,12,22,27,19,9,11,13,42,10,7,16,21,7,17,14,15,20,28,7,11,23,34,24,37,17,7,19,11,8,15
,37,13,18,29,41,7,32,25,19,24,7,28,7,19,15,16,16,17,23,8,23,34,9,15,44,9,18,10,7,49,15,14,18,9,11,15,22,17,28,7,22,53,11,55,34,11,30,28,38,7,8,27,9,11,16,8,9,38,37,18,7,20,3
,3,26,12,16,17,58,14,16,28,10,34,9,29,8,7,13,13,25,7,21,7,19,16,13,28,9,19,42,52,36,15,32,18,17,8,10,18,8,23,17,18,26,15,34,12,13,55,7,16,17,39,7,19,17,64,9,15
,25,13,19,18,13,37,7,18,21,15,7,14,12,16,24,23,17,9,8,11,9,39,16,14,16,35,11,10,19,7,8,8,18,27,27,20,8,22,29,20,7,18,8,39,8,13,9,7,15,16,7,8,16,47,22,18,7,11,7,8,15,9,21,21
,58,41,9,19,23,15,11,35,8,8,17,22,9,16,18,15,12,7,15,15,9,26,22,11,38,7,68,11,17,16,8,18,21,22,27,22,15,7,27,36,8,23,18,26,12,21,11,38,9,28,29,21
,17,9,7,18,10,20,8,25,15,25,15,48,13,19,23,19,7,18,20,9,12,22,30,7,13,36,29,9,29,26,10,17,25,14,7,18,25,10,18,20,12,26,43,10,21,20,14,9,4,22,22,32
,19,15,53,42,11,20,17,13,16,9,7,34,37,12,18,10,24,28,12,15,21,22,31,9,12,14,18,9,29,14,10,7,38,21,9,11,13,7,22,17,16,47,15,20,20,15,11,55,7,16,27,16,42,11,32,22,18,51,21
,10,7,13,12,37,52,15,17,32,13,8,11,16,15,12,10,7,8,14,19,10,8,27,18,14,19,11,12,24,14,21,12,22,27,19,9,11,13,42,10,7,16,21,7,17,14,15,20,28,7,11,23,34,24,37,17,7,19,11,8,15
,37,13,18,29,41,7,32,25,19,24,7,28,7,19,15,16,16,17,23,8,23,34,9,15,44,9,18,10,7,49,15,14,18,9,11,15,22,17,28,7,22,53,11,55,34,11,30,28,38,7,8,27,9,11,16,8,9,38,37,18,7,20,3
,3,26,12,16,17,58,14,16,28,10,34,9,29,8,7,13,13,25,7,21,7,19,16,13,28,9,19,42,52,36,15,32,18,17,8,10,18,8,23,17,18,26,15,34,12,13,55,7,16,17,39,7,19,17,64,9,15
,25,13,19,18,13,37,7,18,21,15,7,14,12,16,24,23,17,9,8,11,9,39,16,14,16,35,11,10,19,7,8,8,18,27,27,20,8,22,29,20,7,18,8,39,8,13,9,7,15,16,7,8,16,47,22,18,7,11,7,8,15,9,21,21
,58,41,9,19,23,15,11,35,8,8,17,22,9,16,18,15,12,7,15,15,9,26,22,11,38,7,68,11,17,16,8,18,21,22,27,22,15,7,27,36,8,23,18,26,12,21,11,38,9,28,29,21
,17,9,7,18,10,20,8,25,15,25,15,48,13,19,23,19,7,18,20,9,12,22,30,7,13,36,29,9,29,26,10,17,25,14,7,18,25,10,18,20,12,26,43,10,21,20,14,9,4,22,22,32
,19,15,53,42,11,20,17,13,16,9,7,34,37,12,18,10,24,28,12,15,21,22,31,9,12,14,18,9,29,14,10,7,38,21,9,11,13,7,22,17,16,47,15,20,20,15,11,55,7,16,27,16,42,11,32,22,18,51,21
,10,7,13,12,37,52,15,17,32,13,8,11,16,15,12,10,7,8,14,19,10,8,27,18,14,19,11,12,24,14,21,12,22,27,19,9,11,13,42,10,7,16,21,7,17,14,15,20,28,7,11,23,34,24,37,17,7,19,11,8,15
,37,13,18,29,41,7,32,25,19,24,7,28,7,19,15,16,16,17,23,8,23,34,9,15,44,9,18,10,7,49,15,14,18,9,11,15,22,17,28,7,22,53,11,55,34,11,30,28,38,7,8,27,9,11,16,8,9,38,37,18,7,20,3
,3,26,12,16,17,58,14,16,28,10,34,9,29,8,7,13,13,25,7,21,7,19,16,13,28,9,19,42,52,36,15,32,18,17,8,10,18,8,23,17,18,26,15,34,12,13,55,7,16,17,39,7,19,17,64,9,15
,25,13,19,18,13,37,7,18,21,15,7,14,12,16,24,23,17,9,8,11,9,39,16,14,16,35,11,10,19,7,8,8,18,27,27,20,8,22,29,20,7,18,8,39,8,13,9,7,15,16,7,8,16,47,22,18,7,11,7,8,15,9,21,21
,58,41,9,19,23,15,11,35,8,8,17,22,9,16,18,15,12,7,15,15,9,26,22,11,38,7,68,11,17,16,8,18,21,22,27,22,15,7,27,36,8,23,18,26,12,21,11,38,9,28,29,21
,17,9,7,18,10,20,8,25,15,25,15,48,13,19,23,19,7,18,20,9,12,22,30,7,13,36,29,9,29,26,10,17,25,14,7,18,25,10,18,20,12,26,43,10,21,20,14,9,4,22,22,32
,19,15,53,42,11,20,17,13,16,9,7,34,37,12,18,10,24,28,12,15,21,22,31,9,12,14,18,9,29,14,10,7,38,21,9,11,13,7,22,17,16,47,15,20,20,15,11,55,7,16,27,16,42,11,32,22,18,51,21
,10,7,13,12,37,52,15,17,32,13,8,11,16,15,12,10,7,8,14,19,10,8,27,18,14,19,11,12,24,14,21,12,22,27,19,9,11,13,42,10,7,16,21,7,17,14,15,20,28,7,11,23,34,24,37,17,7,19,11,8,15
,37,13,18,29,41,7,32,25,19,24,7,28,7,19,15,16,16,17,23,8,23,34,9,15
```

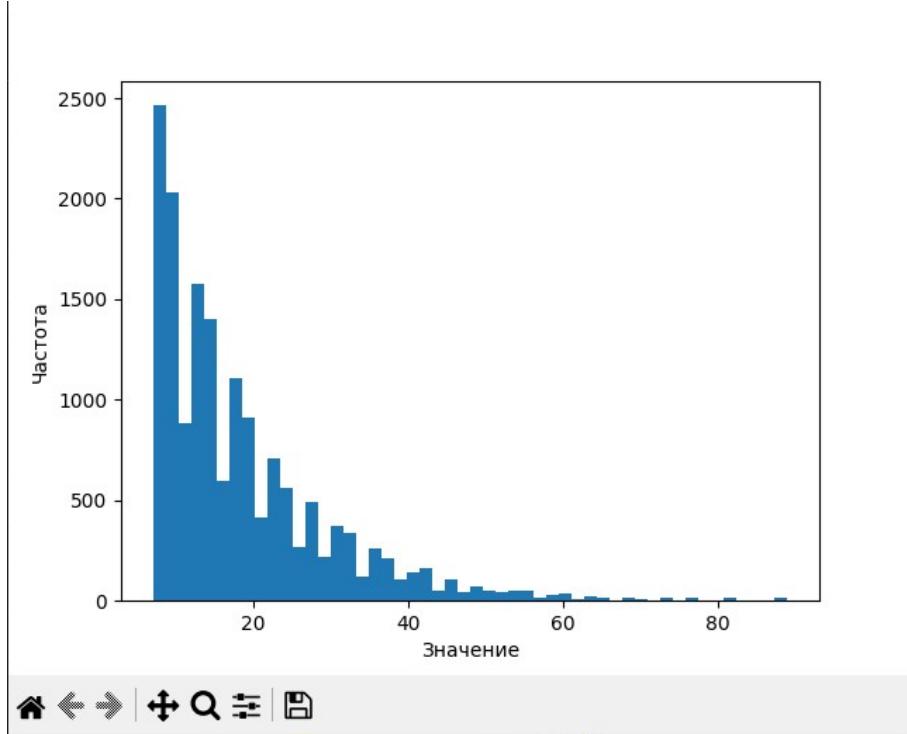


Рисунок 34 – График распределения

Исходный код программы:

```

vector <double> gmFunc(string distributionMethod, string inputFileName, string parameter1,
→   string parameter2, bool defaultVec, bool& errorMessage) {
    ifstream inFile(inputFileName);
    string lineFile;
    vector <double> allElems;
    if (!inFile.is_open()) {
        cout << "Ошибка открытия файла " << inputFileName << endl;
        errorMessage = true;
        return allElems;
    }
    string getlineElem;
    getline(inFile, getlineElem);
    stringstream ss(getlineElem);
    while (getline(ss, getlineElem, ',')) {
        double intElem = stod(getlineElem);
        allElems.push_back(intElem);
    }
    inFile.close();

    vector <double> allU = XtoU(allElems);
    double a = stod(parameter1);
    double b = stod(parameter2);
    for (int i = 0; i < allU.size(); i++)
    {
        if (allU[i] == 1) {

```

```

        cout << "Одно из случайных чисел U = 1. Ошибка в вычислении ln(1 -
        ↳ U)" << endl;
        errorMessage = true;
        return allU;
    }

    allU[i] = a - (b * log(1 - allU[i]));
}

return allU;
}

```

2.6 Логнормальное распределение

Описание алгоритма:

Используя стандартные нормальные случайные числа Z , применяют формулу

$$Y = a + \exp(b - Z)$$

для получения случайных чисел, соответствующих логнормальному распределению.

Параметры запуска программы:

rnc.exe /d:ln /f:D:prngTestMt.txt /p1:2 /p2:1,1

```
D:\cpp_projects\rnc>rnc.exe /d:ln /f:D:prngTestMt.txt /p1:2 /p2:1,1
Distribution method: ln
Parameter 1: 2
Parameter 2: 1,1
Input File: D:prngTestMt.txt
Output File: lnOutput.dat
```

Рисунок 35 – Запуск программы

Рисунок 36 – Последовательность с заданным распределением

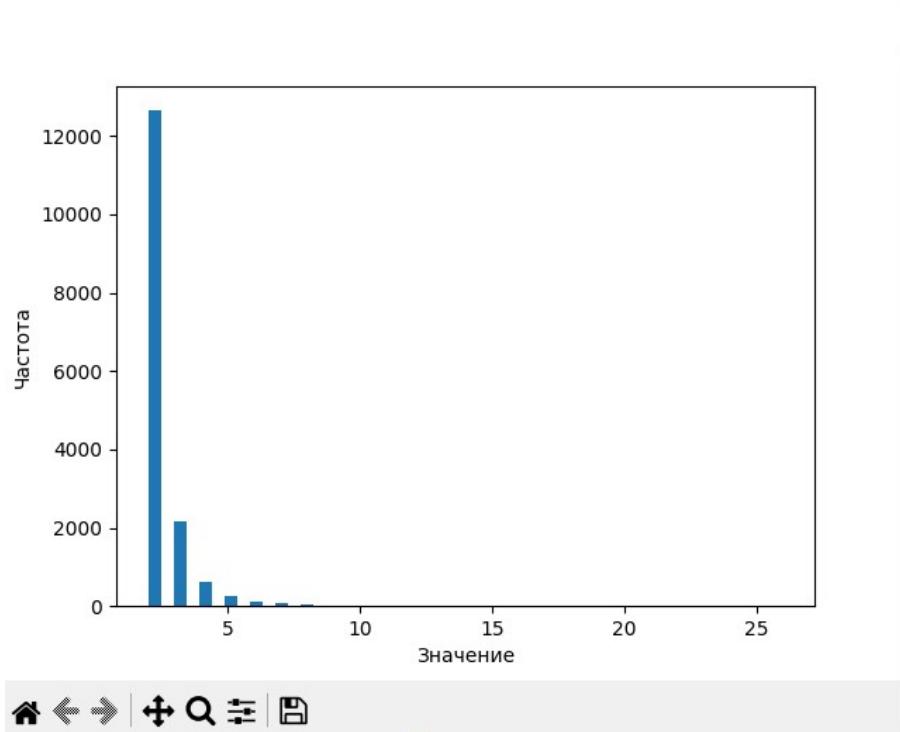


Рисунок 37 – График распределения

Исходный код программы:

```
vector <double> lnFunc(string distributionMethod, string inputFileName, string parameter1,  
→   string parameter2, bool defaultVec, bool& errorMessage) {  
    ifstream inFile(inputFileName);  
    string lineFile;
```

```

vector <double> allElems;
if (!inFile.is_open()) {
    cout << "Ошибка открытия файла " << inputFileName << endl;
    errorMessage = true;
    return allElems;
}
string getlineElem;
getline(inFile, getlineElem);
stringstream ss(getlineElem);
while (getline(ss, getlineElem, ',')) {
    double intElem = stod(getlineElem);
    allElems.push_back(intElem);
}
inFile.close();

if (allElems.size() % 2 != 0)
allElems.erase(allElems.end() - 1);

vector <double> allU = XtoU(allElems);
double a = stod(parameter1);
double b = stod(parameter2);
for (int i = 0; i < allU.size(); i += 2)
{
    if (allU[i] == 1) {
        cout << "Одно из случайных чисел U = 1. Ошибка в вычислении ln(1 -
        ↪ U)" << endl;
        errorMessage = true;
        return allU;
    }
    double U1 = allU[i];
    double U2 = allU[i + 1];
    double Z1 = a + (b * sqrt(-2 * log(1 - U1)) * cos(2 * M_PI * U2));
    double Z2 = a + (b * sqrt(-2 * log(1 - U1)) * sin(2 * M_PI * U2));
    double bminZ1 = b - Z1;
    if (bminZ1 > 43) {
        cout << "Разность b - Z1 больше 43. Выход за границы double в
        ↪ вычислении exp(b - Z1)" << endl;
        errorMessage = true;
        return allU;
    }
    double bminZ2 = b - Z2;
    if (bminZ1 > 43) {
        cout << "Разность b - Z2 больше 43. Выход за границы double в
        ↪ вычислении exp(b - Z2)" << endl;
        errorMessage = true;
        return allU;
    }
    allU[i] = a + exp(bminZ1);
}

```

```
    allU[i + 1] = a + exp(bminZ2);  
}  
  
return allU;  
}
```

2.7 Логистическое распределение

Описание алгоритма:

Если стандартные равномерные случайные числа U генерированы методом, изложенным выше, то случайные числа, соответствующие логистическому распределению, получают по формуле:

$$Y = a + b \ln\left(\frac{U}{1-U}\right)$$

Параметры запуска программы:

rnc.exe /d:ls /f:D:prngTestMt.txt /p1:7 /p2:16

```
D:\cpp_projects\rnc>rnc.exe /d:ls /f:D:prngTestMt.txt /p1:7 /p2:16

Distribution method: ls
Parameter 1: 7
Parameter 2: 16
Input File: D:prngTestMt.txt
Output File: lsOutput.dat
```

Рисунок 38 – Запуск программы

Рисунок 39 – Последовательность с заданным распределением

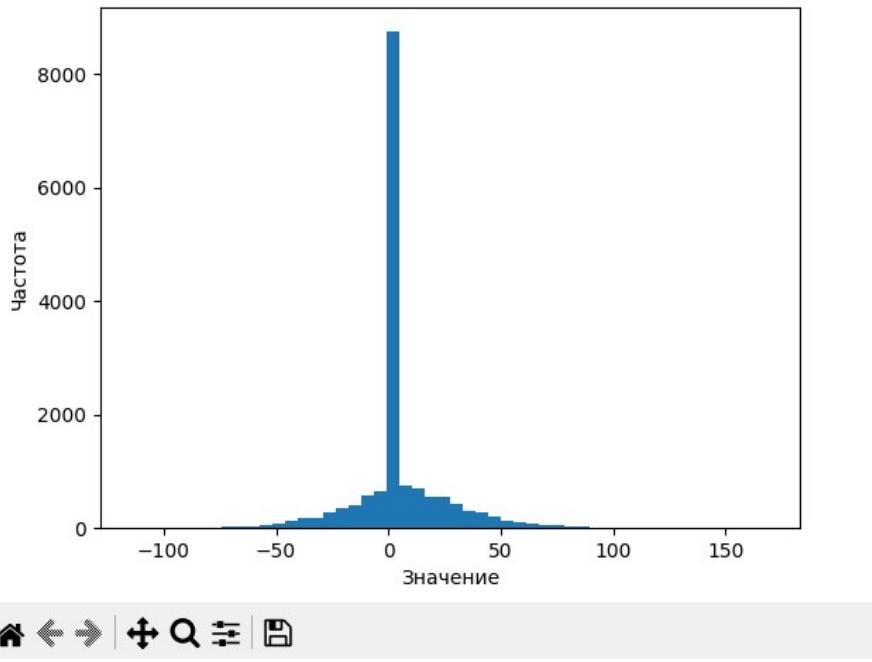


Рисунок 40 – График распределения

Исходный код программы:

```

vector <double> lsFunc(string distributionMethod, string inputFileName, string parameter1,
→  string parameter2, bool defaultVec, bool& errorMessage) {
    ifstream inFile(inputFileName);
    string lineFile;
    vector <double> allElems;
    if (!inFile.is_open()) {
        cout << "Ошибка открытия файла " << inputFileName << endl;
        errorMessage = true;
        return allElems;
    }
    string getlineElem;
    getline(inFile, getlineElem);
    stringstream ss(getlineElem);
    while (getline(ss, getlineElem, ',')) {
        double intElem = stod(getlineElem);
        allElems.push_back(intElem);
    }
    inFile.close();

    vector <double> allU = XtoU(allElems);
    double a = stod(parameter1);
    double b = stod(parameter2);
    for (int i = 0; i < allU.size(); i += 2)
    {
        if (allU[i] == 1) {

```

```

        cout << "Одно из случайных чисел U = 1. Ошибка в вычислении U/(1 -
        ↪ U). Деление на 0." << endl;
        errorMessage = true;
        return allU;
    }

    else if (allU[i] == 0) {
        cout << "Одно из случайных чисел U = 0. Ошибка в вычислении ln(U/(1
        ↪ - U))" << endl;
        errorMessage = true;
        return allU;
    }
    allU[i] = a + (b * log(allU[i] / (1 - allU[i])));
}

return allU;
}

```

2.8 Биномиальное распределение

Описание алгоритма:

Вычисляют функцию распределения

$$F(y) = \sum_{k=0}^y C_n^k p^k (1-p)^{n-k}, y = 0, 1, \dots, n$$

Для получения случайного числа Y генерируют стандартное равномерное случайное число U . Случайное число Y является наименьшим значением y , для которого $U \leq F(y)$.

Параметры запуска программы:

`rnc.exe /d:bi /f:D:prngTestMt.txt /p1:10 /p2:0,5`

```
D:\cpp_projects\rnc>rnc.exe /d:bi /f:D:prngTestMt.txt /p1:10 /p2:0,5
Distribution method: bi
Parameter 1: 10
Parameter 2: 0,5
Input File: D:prngTestMt.txt
Output File: biOutput.dat
```

Рисунок 41 – Запуск программы

Рисунок 42 – Последовательность с заданным распределением

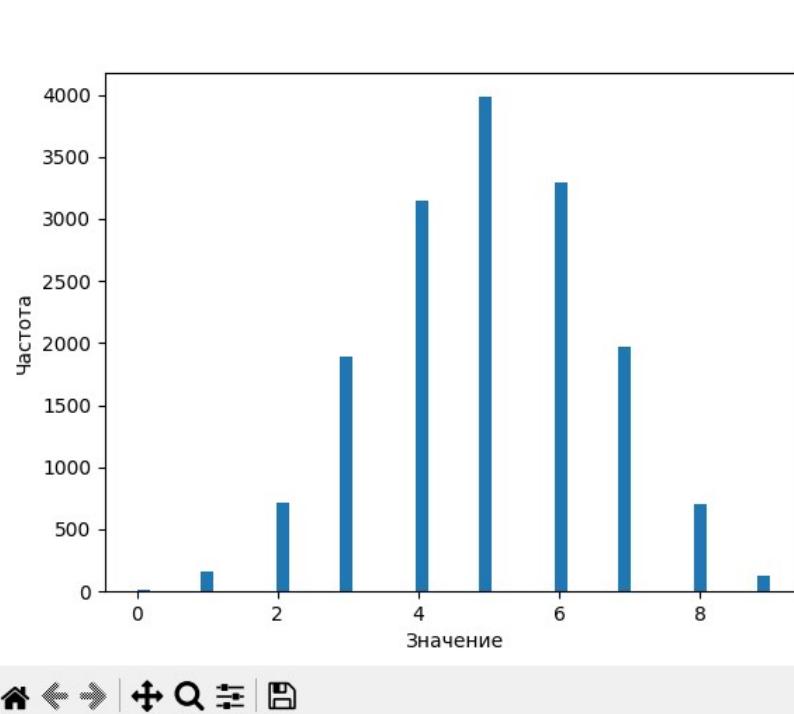


Рисунок 43 – График распределения

Исходный код программы:

```
double Cnk(double n, double k) {  
    double nfact = 1;  
    for (int i = 1; i <= n; i++) {  
        nfact = nfact * i;
```

```

    }

    double nminkfact = 1;
    for (int i = 1; i <= n - k; i++) {
        nminkfact = nminkfact * i;
    }
    double kfact = 1;
    for (int i = 1; i <= k; i++) {
        kfact = kfact * i;
    }
    double res = nfact / (nminkfact * kfact);
    return res;
}

double funcBiF(double y, double n, double p) {
    double resSum = 0;
    for (int k = 0; k <= y; k++)
    {
        resSum = resSum + (Cnk(n, k) * pow(p, k) * pow(1 - p, n - k));
    }
    return resSum;
}

vector <double> biFunc(string distributionMethod, string inputFileName, string parameter1,
→ string parameter2, bool defaultVec, bool& errorMessage) {
    ifstream inFile(inputFileName);
    string lineElem;
    vector <double> allElems;
    if (!inFile.is_open()) {
        cout << "Ошибка открытия файла " << inputFileName << endl;
        errorMessage = true;
        return allElems;
    }
    string getlineElem;
    getline(inFile, getlineElem);
    stringstream ss(getlineElem);
    while (getline(ss, getlineElem, ',')) {
        double intElem = stod(getlineElem);
        allElems.push_back(intElem);
    }
    inFile.close();

    vector <double> allU = XtoU(allElems);
    double n = stod(parameter1); //n
    double p = stod(parameter2); //p
    if (p > 1 || p < 0) {
        cout << "Ошибка. Параметр p2 не может быть > 1 или < 0, т.к. это
→ вероятность." << endl;
        errorMessage = true;
    }
}

```

```

        return allElems;
    }

    for (int i = 0; i < allU.size(); i++)
    {
        double y = 0;
        for (int j = 0; j <= n; j++)
        {
            if (allU[i] > funcBiF(y, n, p)) {
                y++;
                if (j == n) {
                    cout << "Ошибка. Не нашлось такого y = (0,...,n),
                           при котором U <= F(y)" << endl;
                    errorMessage = true;
                    return allElems;
                }
            }
            else
                break;
        }
        allU[i] = y;
    }

    return allU;
}

```

ПРИЛОЖЕНИЕ А

Исходный код задания 1

```
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include <iomanip>
#include <bitset>
#include <sstream>
#include <deque>
#include <cmath>

using namespace std;

deque<bool> binary(int x0, int num_bits) {
    bitset<sizeof(int) * 8> bits(x0);
    deque<bool> binary_digits;
    for (int i = num_bits - 1; i >= 0; --i) {
        binary_digits.push_back(bits[i]);
    }
    return binary_digits;
}

long long binToDec(string binStr)
{
    short i;
    long long res = 0;
    for (i = 0; i < sizeof(long long) * 8 && binStr[i]; ++i)
    {
        res <= 1;
        res |= (binStr[i] - '0');
    }
    return res;
}

long long binDequeToDec(deque<bool> binStr)
{
    short i;
    long long res = 0;
    for (i = 0; i < sizeof(long long) * 8 && binStr[i]; ++i)
    {
        res <= 1;
        res |= (binStr[i] - '0');
    }
    return res;
}
```

```

vector<int> splitStr(string str, char symbol)
{
    vector<int> elems;
    int elem;
    stringstream ss(str);
    while (ss >> elem) {
        elems.push_back(elem);
        if (ss.peek() == symbol) {
            ss.ignore();
        }
    }
    return elems;
}

pair<vector<int>, vector <deque <bool>>> splitStrFsr(string str, char symbol, string
↪ codeMethod)
{
    vector<int> elems;
    vector <deque <bool>> nfsrRegs;
    if (codeMethod == "lfsr") {
        char elem;
        int k = 0;
        vector<string> elemsLfsr(2);
        stringstream ss(str);
        while (ss >> elem) {
            elemsLfsr[k].push_back(elem);
            if (ss.peek() == symbol) {
                ss.ignore();
                k++;
            }
        }
        elems.push_back(stoi(elemsLfsr[1]));
        int elemsLfsrSize = elemsLfsr[0].size();
        for (int i = 0; i < elemsLfsrSize; i++)
        {
            if (elemsLfsr[0][i] == '1')
            {
                elems.push_back(elemsLfsrSize - 1 - i);
            }
        }
        //x0,coeffs
    }
    else {
        char elem;
        int k = 0;
        vector<string> elemsNfsr(3);
        stringstream ss(str);
        while (ss >> elem) {

```

```

        elemsNfsr[k].push_back(elem);
        if (ss.peek() == symbol) {
            ss.ignore();
            k++;
        }
    }
    for (int j = 0; j < elemsNfsr.size(); j++)
    {
        int elemsNfsrSize = elemsNfsr[j].size();
        int coeffsCount = 0;
        vector <int> elemsDop;
        for (int i = 0; i < elemsNfsrSize; i++)
        {
            if (elemsNfsr[j][i] == '1')
            {
                elemsDop.push_back(elemsNfsrSize - 1 - i);
                coeffsCount++;
            }
        }
        elems.push_back(coeffsCount);
        for (int i = 0; i < coeffsCount; i++)
        {
            elems.push_back(elemsDop[i]);
        }
    }
    //size1,coeffs1,size2,coeffs2,size3,coeffs3
    for (int i = 0; i < 3; i++)
    {
        deque<bool> strInDeq;
        for (int j = 0; j < elemsNfsr[i].size(); j++)
        {
            if (elemsNfsr[i][j] == '1')
                strInDeq.push_back(1);
            else
                strInDeq.push_back(0);
        }
        nfsrRegs.push_back(strInDeq);
    }
    pair<vector<int>, vector <deque <bool>>> resPair = { elems , nfsrRegs };
    return resPair;
}

deque<bool> multRs(deque<bool> r1, deque<bool> r2) {
    int r1size = r1.size();
    int r2size = r2.size();
    if (r1size > r2size) {
        while (r2size != 0) {

```

```

        r1[r1size] &= r2[r2size];
        r2size--;
        r1size--;
    }
    return r1;
}
else {
    while (r1size != 0) {
        r1[r2size] &= r2[r1size];
        r2size--;
        r1size--;
    }
    return r2;
}
return r1;
}

deque<bool> orRs(deque<bool> r1, deque<bool> r2) {
    int r1size = r1.size();
    int r2size = r2.size();
    if (r1size > r2size) {
        while (r2size != 0) {
            r1[r1size] |= r2[r2size];
            r2size--;
            r1size--;
        }
        return r1;
    }
    else {
        while (r1size != 0) {
            r1[r2size] |= r2[r1size];
            r2size--;
            r1size--;
        }
        return r2;
    }
    return r1;
}

deque<bool> xorRs(deque<bool> r1, deque<bool> r2) {
    int r1size = r1.size();
    int r2size = r2.size();
    if (r1size > r2size) {
        while (r2size != 0) {
            r1[r1size] ^= r2[r2size];
            r2size--;
            r1size--;
        }
    }
}

```

```

        return r1;
    }
    else {
        while (r1size != 0) {
            r1[r2size] ^= r2[r1size];
            r2size--;
            r1size--;
        }
        return r2;
    }
    return r1;
}

string findInStr(string const& str, int n) {
    if (str.length() < n) {
        return str;
    }
    return str.substr(0, n);
}

void dequePrint(deque<bool> reg) {
    cout << "deque: {";
    int regSize = reg.size();
    for (int i = 0; i < regSize; i++) {
        if (i != regSize - 1)
            cout << reg[i] << ",";
        else
            cout << reg[i] << "}";
    }
    cout << ";" size: " << regSize << "." << endl;
    return;
}

int powmod(int x, int e, int n) {
    for (int i = 0; i < e - 1; i++)
    {
        x = (x * x) % n;
    }
    return x;
}

long long nod(long long x, long long y) {
    while (x != y) {
        if (x > y) {
            x = x - y;
        }
        else {

```

```

        y = y - x;
    }
}

return x;
}

bool mal_fermaPrime(long long p) {
    vector <long long> test;
    long long prime = 0;
    long long not_prime = 0;
    for (long long i = 1; i < 100; i++)
    {
        long long pi = p - i;
        if (pi >= 2 && nod(pi, p) == 1)
        {
            test.push_back(pi);
        }
    }
    for (long long i = 0; i < test.size(); i++)
    {
        long long a_step = test[i];
        long long a_step_dop = test[i];
        for (int j = 1; j < p - 1; j++)
        {
            a_step *= a_step_dop;
            a_step = a_step % p;
        }
        if (a_step == 1) {
            prime++;
        }
        else {
            not_prime++;
        }
    }
    if (prime <= not_prime)
        return false;
    else
        return true;
}

bool kratP(long long a, long long b) {
    for (int i = 2; i < b; i++)
    {
        if (b % i == 0 && a % i == 0 && mal_fermaPrime(i))
            return true;
    }
    return false;
}

```

```

vector<int> progressVector(int NumCount) {
    vector<int> shows;
    int NumCount10 = NumCount / 10;
    for (int i = 1; i < 10; i++)
    {
        shows.push_back(NumCount10 * i);
    }
    shows.push_back(NumCount - 1);
    cout << "\nThe process of generating pseudo-random numbers: \n\n";
    return shows;
}

deque <bool> gen_p_lc(int p) {
    deque <bool> ps;
    //Модуль, множитель, приращение, начальное значение
    long long m = 7875;
    long long a = 421;
    long long c = 1663;
    long long x0 = 32112;

    for (int i = 0; i < p; i++)
    {
        x0 = (x0 * a + c) % m;
        ps.push_back(x0 % 2);
    }
    return ps;
}

void helpFunc() {
    cout << "The command with /h is introduced. Permissible parameters:";
    cout << "\n\n/g:<method_code> - the parameter specifies the method of DRB
    generation, where the method_code can be one of the following:\n";
    cout << "\n lc - linear congruent method (Input: module, multiplier,
    increment, initial value);\n add - additive method (Input: module, low
    index, high index, sequence of initial values);\n 5p - five-parameter
    method (Input: p, q1, q2, q3, w);\n lfsr - Linear-feedback shift
    register (LFSR) (Input: binary representation of the coefficient vector,
    initial value of the register);\n nfsr - non-linear combination of LFSR
    (Input: binary representation of the coefficient vectors for R1, R2,
    R3);\n mt - Mersenne vortex (Input: module, initial value of x);\n rc4 -
    RC4 (Input: 256 initial values);\n rsa - RSA-based PRNG (Input: module
    n, number e, initial value of x; e satisfies conditions: 1 < e <
    (p-1)(q-1), NOD(e, (p-1)(q-1)) = 1, where p*q=n. x from the interval
    [1,n]);\n bbs - Blum-Blum-Schub algorithm (Input: Initial value of x
    (mutually prime with n));\n";
    cout << "\n\n/i:<number> - initialization vector of the generator.";
```

```

        cout << "\n\n/n:<length> - the number of generated numbers. If the parameter
        ↵  is not specified - 10000 numbers are generated.";
        cout << "\n\n/f:<full_file_name> - full name of the file in which the data
        ↵  will be output. If the parameter is not specified, the data should be
        ↵  written to the file named rnd.dat.";
        cout << "\n\n/h - information about the allowed command line parameters of
        ↵  the program.\n";

    }

// Linear congruent method
void lcFunc(string codeMethod, vector <int> genVec, int NumCount, const char*
        ↵  fileName, bool defaultVec) {
    long long m = 12960;
    long long a = 1741;
    long long c = 2731;
    long long x0 = 1;
    if (!defaultVec) {
        m = genVec[0];
        a = genVec[1];
        c = genVec[2];
        x0 = genVec[3];
    }
    if (m <= 0 || a > m || a < 0 || c > m || c < 0 || x0 > m || x0 < 0) {
        cout << "\nError. The parameters do not meet the requirements.\n";
        return;
    }
    if (nod(c, m) != 1) {
        cout << "\nThe parameters do not meet the requirements of Theorem
        ↵  3.1 (Item 1).\n";
    }
    if (!(kratP(a - 1, m))) {
        cout << "\nThe parameters do not meet the requirements of Theorem
        ↵  3.1 (Item 2).\n";
    }
    if (m % 4 == 0 && (a - 1) % 4 != 0) {
        cout << "\nThe parameters do not meet the requirements of Theorem 3.1 (Item
        ↵  3).\n";
    }
    vector<int> showsVector = progressVector(NumCount);
    int check_progress_index = 0;

    ofstream f;
    f.open(fileName, ios::out);
    for (int i = 0; i < NumCount; i++)
    {
        x0 = (x0 * a + c) % m;
        if (defaultVec)

```

```

        f << x0 % 1001 << ",";

    else
        f << x0 << ",";

    if (i == showsVector[check_progress_index]) {
        cout << (check_progress_index + 1) * 10 << "% completed\n";
        check_progress_index++;
    }
}
f.close();
return;
}

// Additive method
void addFunc(string codeMethod, vector <int> genVec, int NumCount, const char* fileName,
→ bool defaultVec) {
// Input: module, low index, high index, sequence of initial values
long long m = 8001;
long long k = 18;
long long j = 65;
long long x0;
vector <long long> xNs;
if (!defaultVec) {
    m = genVec[0];
    k = genVec[1];
    j = genVec[2];
    for (int i = 3; i < genVec.size(); i++)
    {
        xNs.push_back(genVec[i]);
    }
}
else {
    xNs = { 816, 159, 798, 290, 168, 441, 691, 655, 874, 220, 125, 977, 586, 381, 868,
    → 294, 948, 437, 581, 181, 701, 536, 11, 672, 103, 601, 794, 189, 12, 130, 386,
    → 828, 288, 183, 117, 456, 624, 807, 110, 498, 27, 234, 474, 613, 615, 341, 906,
    → 562, 778, 486, 155, 276, 894, 441, 226, 762, 234, 762, 98, 458, 399, 445, 765,
    → 223, 879 };
}

if (k < 1 || k >= j || j < 1) {
    cout << "\nError. The parameters do not meet the requirements.\n";
    return;
}

vector<int> showsVector = progressVector(NumCount);
int check_progress_index = 0;

```

```

ofstream f;
f.open(fileName, ios::out);
long long maxkj = max(k, j);
if (xNs.size() < maxkj) {
    cout << "\nError. The parameters do not meet the requirements (The transferred
    ↵ initial values are insufficient).\n";
    return;
}
for (int i = maxkj + 1; i < NumCount + maxkj + 1; i++)
{
    x0 = (xNs[i - k] + xNs[i - j]) % m;
    xNs.push_back(x0);
    if (defaultVec)
        f << x0 % 1001 << ",";
    else
        f << x0 << ",";
}

if (i == showsVector[check_progress_index]) {
    cout << (check_progress_index + 1) * 10 << "% completed\n";
    check_progress_index++;
}
f.close();
return;
}

// Five-parameter method
void fivePFunc(string codeMethod, vector <int> genVec, int NumCount, const char* fileName,
    ↵ bool defaultVec) {
// Input: p, q1, q2, q3, w
int p = 4253;
long long q1 = 1093;
long long q2 = 2254;
long long q3 = 3297;
int w = 16;
if (!defaultVec) {
    p = genVec[0];
    q1 = genVec[1];
    q2 = genVec[2];
    q3 = genVec[3];
    w = genVec[4];
}

if (q1 >= p || q2 >= p || q3 >= p) {
    cout << "\nError. The parameters do not meet the requirements.\n";
    return;
}

```

```

int regLength = p;
deque <bool> reg = gen_p_lc(p);
vector<int> showsVector = progressVector(NumCount);
int check_progress_index = 0;

ofstream f;
f.open(fileName, ios::out);
for (int i = 0; i < NumCount; i++)
{
    bool xNplusP = reg[regLength - q1 - 1] ^ reg[regLength - q2 - 1] ^ reg[regLength -
    ↳ q3 - 1];
    reg.push_front(xNplusP);
    reg.pop_back();
    long long ReginDec = 0;
    for (int h = 0; h < w; h++)
    {
        ReginDec *= 2;
        ReginDec += reg[h];
    }
    if (defaultVec)
        f << ReginDec % 1001 << ",";
    else
        f << ReginDec << ",";
}

if (i == showsVector[check_progress_index]) {
    cout << (check_progress_index + 1) * 10 << "% completed\n";
    check_progress_index++;
}
}

f.close();
return;
}

// Linear-feedback shift register (LFSR)
void lfsrFunc(string codeMethod, vector <int> genVec, int NumCount, const char* fileName,
→ bool defaultVec) {
// Input: binary representation of the coefficient vector, initial value of the register
if (defaultVec) {
    genVec = { 121, 14, 5, 3, 1, 0 }; // x0, coeffs (100000000101011)
}
int x0 = genVec[0];
int x0Length = log2(x0) + 1;
int regLength = x0Length - 1;
bool bigCoeff = false;
for (int i = 1; i < genVec.size(); i++)
{
    if (genVec[i] > regLength) {
        regLength = genVec[i] + 1;
    }
}

```

```

        bigCoeff = true;
    }
}

if (x0 < 0) {
    cout << "\nError. The parameters do not meet the requirements.\n";
    return;
}

vector<int> showsVector = progressVector(NumCount);
int check_progress_index = 0;

deque<bool> reg;
if (!bigCoeff) {
    regLength++;
    reg = binary(x0, regLength);
}
else {
    reg.assign(regLength - x0Length, 0);
    deque<bool> regDop = binary(x0, x0Length);
    for (int i = 0; i < regDop.size(); i++)
    {
        reg.push_back(regDop[i]);
    }
}
ofstream f;
f.open(fileName, ios::out);
for (int i = 0; i < NumCount; i++)
{
    bool xNplusP = reg[regLength - genVec[1] - 1];
    for (int j = 2; j < genVec.size(); j++)
    {
        xNplusP ^= reg[regLength - genVec[j] - 1];
    }
    reg.push_front(xNplusP);
    reg.pop_back();
    long long ReginDec = 0;
    for (int h = 0; h < reg.size(); h++)
    {
        ReginDec *= 2;
        ReginDec += reg[h];
    }
    if (defaultVec)
        f << ReginDec % 1001 << ",";
    else
        f << ReginDec << ",";
}

if (i == showsVector[check_progress_index]) {
    cout << (check_progress_index + 1) * 10 << "% completed\n";
}

```

```

        check_progress_index++;
    }
}

f.close();
return;
}

// Non-linear combination of LFSR
void nfsrFunc(string codeMethod, vector <int> genVec, int NumCount, const char* fileName,
→ vector <deque <bool>> nfsrRegs, bool defaultVec) {
// Input: binary representation of the coefficient vectors for R1, R2, R3
if (defaultVec) {
    genVec = { 5, 14, 4, 3, 1, 0, 3, 15, 1, 0, 5, 16, 5, 3, 2, 0 }; // → size1,coeffs1,size2,coeffs2,size3,coeffs3 → (100000000011011,1000000000000011,10000000000101101)
    nfsrRegs = { {1,0,0,0,0,0,0,0,0,1,1,0,1,1}, {1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1}, → {1,0,0,0,0,0,0,0,0,1,0,1,1,0,1} };
}
int nachR1 = 1;
int endR1 = genVec[0] + 1;
int nachR2 = endR1 + 1;
int endR2 = genVec[endR1] + nachR2 + 1;
int nachR3 = endR2 + 1;
int endR3 = genVec.size();
vector<int> startsRslos = { nachR1, nachR2, nachR3 };
vector<int> endssRslos = { endR1, endR2, endR3 };
vector<int> regLengths;
for (int i = 0; i < 3; i++)
{
    regLengths.push_back(nfsrRegs[i].size());
}
vector <deque<bool>> regs = nfsrRegs;

vector<int> showsVector = progressVector(NumCount);
int check_progress_index = 0;

ofstream f;
f.open(fileName, ios::out);
for (int i = 0; i < NumCount; i++)
{
    for (int j = 0; j < 3; j++)
    {
        int startH = startsRslos[j];
        int endH = endssRslos[j];
        bool xNext = regs[j][regLengths[j] - genVec[startH] - 1];
        for (int h = startH; h < endH; h++)
        {
            xNext ^= regs[j][regLengths[j] - genVec[h] - 1];
        }
    }
}

```



```

if (defaultVec) {
    genVec = { 10001, 8191 };
}
long long zMod = genVec[0];
long long x0 = genVec[1];
vector <long long> x_zns(1, x0);
for (int i = 0; i < p; i++)
{
    x_zns.push_back(abs(1812433253 * (x_zns[i - 1] ^ (x_zns[i - 1] >> 30)) + i));
}

vector<int> showsVector = progressVector(NumCount);
int check_progress_index = 0;

ofstream f;
f.open(fileName, ios::out);
int n = 0;
for (int countN = 0; countN < NumCount; countN++)
{
    long long xn = x_zns[n];
    long long xnPlus1 = x_zns[(n + 1) % p];
    long long yNew = (xn & uDec) | (xnPlus1 & hDec);
    int yNewLength = log2(yNew) + 1;
    deque<bool> yBin = binary(yNew, yNewLength);
    long long Xnew;
    yNew >>= 1;
    long long xnqp = x_zns[(n + q) % p];
    if (yBin[yBin.size() - 1] == 1)
        Xnew = xnqp ^ yNew ^ a;
    else
        Xnew = xnqp ^ yNew ^ 0;
    yNew = Xnew;
    yNew = yNew ^ (yNew >> u);
    yNew = yNew ^ ((yNew << s) & b);
    yNew = yNew ^ ((yNew << t) & c);
    long long zNew = yNew ^ (yNew >> l);
    if (defaultVec)
        f << (zNew % zMod) % 1001 << ",";
    else
        f << zNew % zMod << ",";

    if (countN == showsVector[check_progress_index]) {
        cout << (check_progress_index + 1) * 10 << "% completed\n";
        check_progress_index++;
    }
    x_zns[n] = Xnew;
    n = (n + 1) % p;
}

```

```

f.close();
return;
}

// RC4
void rc4Func(string codeMethod, vector <int> genVec, int NumCount, const char* fileName,
→ bool defaultVec) {
// Input: 256 initial values
if (defaultVec) {
    genVec = { 802, 720, 341, 337, 961, 882, 417, 785, 198, 727, 899, 372, 374, 425,
    → 556, 615, 813, 768, 840, 183, 893, 568, 73, 387, 18, 436, 182, 125, 806, 899,
    → 485, 607, 619, 825, 944, 579, 707, 360, 363, 904, 87, 262, 276, 460, 687, 831,
    → 75, 499, 599, 915, 681, 492, 483, 754, 878, 500, 189, 60, 624, 994, 959, 109,
    → 600, 577, 934, 544, 156, 640, 903, 519, 544, 990, 781, 819, 449, 468, 650, 524,
    → 967, 248, 438, 647, 739, 920, 400, 617, 419, 588, 676, 43, 581, 634, 151, 181,
    → 211, 84, 724, 367, 723, 627, 886, 267, 617, 667, 85, 65, 134, 735, 589, 100,
    → 983, 26, 747, 721, 945, 147, 337, 364, 734, 13, 406, 315, 647, 556, 496, 858,
    → 640, 220, 224, 362, 847, 110, 629, 463, 776, 713, 528, 909, 448, 116, 9, 430,
    → 141, 755, 151, 86, 901, 488, 449, 635, 500, 855, 950, 147, 410, 446, 4, 49, 665,
    → 227, 411, 511, 336, 39, 974, 112, 752, 501, 21, 200, 617, 29, 629, 757, 784,
    → 779, 843, 684, 266, 292, 319, 766, 146, 269, 912, 556, 714, 916, 605, 378, 142,
    → 15, 889, 478, 54, 862, 590, 806, 363, 610, 5, 979, 638, 634, 736, 421, 413, 578,
    → 105, 679, 869, 424, 444, 14, 692, 356, 569, 405, 271, 173, 783, 413, 188, 671,
    → 891, 242, 533, 480, 48, 895, 89, 53, 873, 727, 686, 608, 147, 98, 185, 252, 776,
    → 54, 675, 220, 67, 366, 576, 636, 771, 846, 808, 553, 259, 996, 224, 149 };
}

vector <int> k = genVec;
vector <int> s = genVec;
int j = 0;
for (int i = 0; i < 256; i++)
{
    j = (j + s[i] + k[i]) % 256;
    swap(s[i], s[j]);
}
int i = 0;
j = 0;

vector<int> showsVector = progressVector(NumCount);
int check_progress_index = 0;

ofstream f;
f.open(fileName, ios::out);
for (int countN = 0; countN < NumCount; countN++)
{
    i = (i + 1) % 256;
    j = (j + s[j]) % 256;
    swap(s[i], s[j]);
    int t = (s[i] + s[j]) % 256;
}

```

```

        if (defaultVec)
            f << s[t] % 1001 << "," ;
        else
            f << s[t] << "," ;

        if (countN == showsVector[check_progress_index]) {
            cout << (check_progress_index + 1) * 10 << "% completed\n";
            check_progress_index++;
        }
    }
    f.close();
    return;
}

// RSA-based PRNG
void rsaFunc(string codeMethod, vector <int> genVec, int NumCount, const char* fileName,
→ bool defaultVec) {
// Input: module n, number e, initial value of x; e satisfies conditions: 1 < e <
→ (p-1)(q-1), NOD(e, (p-1)(q-1)) = 1, where p*q=n. x from the interval [1,n]
if (defaultVec) {
    genVec = { 7191817, 151, 69 };
}
int n = genVec[0];
int e = genVec[1];
int x = genVec[2];
int l = 20;

if (e <= 1 || x < 1 || x > n - 1) {
    cout << "\nError. The parameters do not meet the requirements.\n";
    return;
}

vector<int> showsVector = progressVector(NumCount);
int check_progress_index = 0;

ofstream f;
f.open(fileName, ios::out);
for (int countN = 0; countN < NumCount; countN++)
{
    string zs = "";
    for (int i = 0; i < l; i++)
    {
        x = powmod(x, e, n);
        char z = '0';
        if (x % 2 != 0)
            z = '1';
        zs.push_back(z);
    }
}

```

```

        if (defaultVec)
            f << binToDec(zs) % 1001 << ",";
        else
            f << binToDec(zs) << ",";

        if (countN == showsVector[check_progress_index]) {
            cout << (check_progress_index + 1) * 10 << "% completed\n";
            check_progress_index++;
        }
    }
    f.close();
    return;
}

// Blum-Blum-Schub algorithm
void bbsFunc(string codeMethod, vector <int> genVec, int NumCount, const char* fileName,
             bool defaultVec) {
    // Input: Initial value of x (mutually prime with n)
    if (defaultVec) {
        genVec = { 8627 };
    }
    int x0 = genVec[0];
    int n = 16637;
    int l = 10;

    if (nod(x0, n) != 1) {
        cout << "\nThe parameters do not meet the requirements (x and n are not mutually
             prime).\n";
    }
}

vector<int> showsVector = progressVector(NumCount);
int check_progress_index = 0;

ofstream f;
f.open(fileName, ios::out);
for (int countN = 0; countN < NumCount; countN++)
{
    x0 = (x0 * x0) % n;
    string zs = "";
    for (int i = 0; i < l; i++)
    {
        x0 = (x0 * x0) % n;
        char z = '0';
        if (x0 % 2 != 0)
            z = '1';
        zs.push_back(z);
    }
    if (defaultVec)

```

```

        f << binToDec(zs) % 1001 << ",";
    else
        f << binToDec(zs) << ",";

    if (countN == showsVector[check_progress_index]) {
        cout << (check_progress_index + 1) * 10 << "% completed\n";
        check_progress_index++;
    }
}

f.close();
return;
}

void setMethod(string codeMethod, vector <int> genVec, int NumCount, const char* fileName,
    → int genVecSize, vector <deque <bool>> nfsrRegs, bool defaultVec) {
if (codeMethod == "lc")
    lcFunc(codeMethod, genVec, NumCount, fileName, defaultVec);
else if (codeMethod == "add")
    addFunc(codeMethod, genVec, NumCount, fileName, defaultVec);
else if (codeMethod == "5p")
    fivePFunc(codeMethod, genVec, NumCount, fileName, defaultVec);
else if (codeMethod == "lfsr")
    lfsrFunc(codeMethod, genVec, NumCount, fileName, defaultVec);
else if (codeMethod == "nfsr")
    nfsrFunc(codeMethod, genVec, NumCount, fileName, nfsrRegs, defaultVec);
else if (codeMethod == "mt")
    mtFunc(codeMethod, genVec, NumCount, fileName, defaultVec);
else if (codeMethod == "rc4")
    rc4Func(codeMethod, genVec, NumCount, fileName, defaultVec);
else if (codeMethod == "rsa")
    rsaFunc(codeMethod, genVec, NumCount, fileName, defaultVec);
else if (codeMethod == "bbs")
    bbsFunc(codeMethod, genVec, NumCount, fileName, defaultVec);
}

int main(int argc, char* argv[])
{
    setlocale(LC_ALL, "Rus");
    string codeMethod;
    vector <int> genVec;
    vector <deque <bool>> nfsrRegs;
    int NumCount = 10000;
    string fName = "rnd.dat";
    bool defaultVec = true;
    for (int i = 0; argv[i]; i++)
    {
        string checkStr = string(argv[i]);
        if (findInStr(checkStr, 2) == "/h") {

```

```

    helpFunc();
    return 0;
}
if (checkStr.length() > 2) {
    string ifStr = findInStr(checkStr, 3);
    string subStr = checkStr.substr(3, checkStr.length());
    if (ifStr == "/g:") {
        codeMethod = subStr;
    }
    if (ifStr == "/i:" && (codeMethod == "lfsr" || codeMethod == "nfsr")) {
        char symbol = ',';
        pair<vector<int>, vector<deque<bool>>> nfsrPair;
        nfsrPair = splitStrFsr(subStr, symbol, codeMethod);
        genVec = nfsrPair.first;
        nfsrRegs = nfsrPair.second;
        defaultVec = false;
    }
    else if (ifStr == "/i:") {
        char symbol = ',';
        genVec = splitStr(subStr, symbol);
        defaultVec = false;
    }
    if (ifStr == "/n:") {
        NumCount = stoi(subStr);
    }
    if (ifStr == "/f:") {
        fName = subStr;
    }
}
const char* fileName = fName.c_str();
int genVecSize = genVec.size();
setMethod(codeMethod, genVec, NumCount, fileName, genVecSize, nfsrRegs, defaultVec);
cout << "\nGeneration method: " << codeMethod << endl;
cout << "Count of generated numbers: " << NumCount << endl;
cout << "Full name of the output file: " << fileName << endl;
return 0;
}

```

ПРИЛОЖЕНИЕ Б

Исходный код задания 2

```
#define _USE_MATH_DEFINES

#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include <iomanip>
#include <bitset>
#include <sstream>
#include <deque>
#include <cmath>

using namespace std;

string findInStr(string const& str, int n) {
    if (str.length() < n) {
        return str;
    }
    return str.substr(0, n);
}

void helpFunc() {
    cout << "Введена команда с /h. Допустимые параметры:";
    cout << "\n\n/d:<распределение> - код распределения для преобразования
    → последовательности. Возможные коды распределений:\n";
    cout << "\n st - стандартное равномерное с заданным интервалом (Параметры: a, b);\n
    → tr - треугольное распределение (Параметры: a, b);\n ex - общее экспоненциальное
    → распределение (Параметры: a, b);\n nr - нормальное распределение (Параметры:
    → mu, sigma);\n gm - гамма распределение (Параметры: a, b);\n ln - логнормальное
    → распределение (Параметры: a, b);\n ls - логистическое распределение (Параметры:
    → a, b);\n bi - биномиальное распределение. (Параметры: n, p)\n";
    cout << "\n\n/f:<имя_файла> - имя файла с входной последовательностью";
    cout << "\n\n/p1:<параметр1> - 1-й параметр, необходимый, для генерации ПСЧ
    → заданного распределения.";
    cout << "\n\n/p2:<параметр2> - 2-й параметр, необходимый, для генерации ПСЧ
    → заданного распределения.";
    cout << "\n\n/h - информация о допустимых параметрах командной строки программы.\n";
}

vector <double> XtoU(vector <double> allElems) {
    int allElemsSize = allElems.size();
    double max = 2.22507e-308;
    for (int i = 0; i < allElemsSize; i++)
    {
        if (allElems[i] > max)
```

```

        max = allElems[i];
    }
    max++;
    for (int i = 0; i < allElemsSize; i++)
    {
        allElems[i] = allElems[i] / max;
    }
    return allElems;
}

vector <double> stFunc(string distributionMethod, string inputFileName, string parameter1,
→ string parameter2, bool defaultVec, bool& errorMessage) {
    ifstream inFile(inputFileName);
    string lineFile;
    vector <double> allElems;
    if (!inFile.is_open()) {
        cout << "Ошибка открытия файла " << inputFileName << endl;
        errorMessage = true;
        return allElems;
    }
    string getlineElem;
    getline(inFile, getlineElem);
    stringstream ss(getlineElem);
    while (getline(ss, getlineElem, ',')) {
        double intElem = stod(getlineElem);
        allElems.push_back(intElem);
    }
    inFile.close();

    vector <double> allU = XtoU(allElems);
    double a = stod(parameter1);
    double b = stod(parameter2);
    for (int i = 0; i < allU.size(); i++)
    {
        allU[i] = (b * allU[i]) + a;
    }

    return allU;
}

vector <double> trFunc(string distributionMethod, string inputFileName, string parameter1,
→ string parameter2, bool defaultVec, bool& errorMessage) {
    ifstream inFile(inputFileName);
    string lineFile;
    vector <double> allElems;
    if (!inFile.is_open()) {
        cout << "Ошибка открытия файла " << inputFileName << endl;
        errorMessage = true;

```

```

        return allElems;
    }

    string getlineElem;
    getline(inFile, getlineElem);
    stringstream ss(getlineElem);
    while (getline(ss, getlineElem, ',')) {
        double intElem = stod(getlineElem);
        allElems.push_back(intElem);
    }
    inFile.close();

    vector <double> allU = XtoU(allElems);
    double a = stod(parameter1);
    double b = stod(parameter2);
    for (int i = 0; i < allU.size() - 1; i++)
    {
        allU[i] = a + (b * (allU[i] + allU[i + 1] - 1));
    }

    return allU;
}

vector <double> exFunc(string distributionMethod, string inputFileName, string parameter1,
→ string parameter2, bool defaultVec, bool& errorMessage) {
    ifstream inFile(inputFileName);
    string lineFile;
    vector <double> allElems;
    if (!inFile.is_open()) {
        cout << "Ошибка открытия файла " << inputFileName << endl;
        errorMessage = true;
        return allElems;
    }
    string getlineElem;
    getline(inFile, getlineElem);
    stringstream ss(getlineElem);
    while (getline(ss, getlineElem, ',')) {
        double intElem = stod(getlineElem);
        allElems.push_back(intElem);
    }
    inFile.close();

    vector <double> allU = XtoU(allElems);
    double a = stod(parameter1);
    double b = stod(parameter2);
    for (int i = 0; i < allU.size() - 1; i++)
    {
        if (allU[i] == 0) {

```

```

        cout << "Одно из случайных чисел U = 0. Ошибка в вычислении ln(U)"
        << endl;
        errorMessage = true;
        return allU;
    }

    allU[i] = ((b * (-1)) * log(allU[i])) + a;
}

return allU;
}

vector <double> nrFunc(string distributionMethod, string inputFileName, string parameter1,
→ string parameter2, bool defaultVec, bool& errorMessage) {
    ifstream inFile(inputFileName);
    string lineFile;
    vector <double> allElems;
    if (!inFile.is_open()) {
        cout << "Ошибка открытия файла " << inputFileName << endl;
        errorMessage = true;
        return allElems;
    }
    string getlineElem;
    getline(inFile, getlineElem);
    stringstream ss(getlineElem);
    while (getline(ss, getlineElem, ',')) {
        double intElem = stod(getlineElem);
        allElems.push_back(intElem);
    }
    inFile.close();

    if (allElems.size() % 2 != 0)
        allElems.erase(allElems.end() - 1);

    vector <double> allU = XtoU(allElems);
    double mu = stod(parameter1);
    double sigma = stod(parameter2);
    for (int i = 0; i < allU.size(); i += 2)
    {
        if (allU[i] == 1) {
            cout << "Одно из случайных чисел U = 1. Ошибка в вычислении ln(1 -
            → U)" << endl;
            errorMessage = true;
            return allU;
        }
        double U1 = allU[i];
        double U2 = allU[i + 1];
        allU[i] = mu + (sigma * sqrt(-2 * log(1 - U1)) * cos(2 * M_PI * U2)); // Z1
    }
}

```

```

        allU[i + 1] = mu + (sigma * sqrt(-2 * log(1 - U1)) * sin(2 * M_PI * U2)); // ← Z2
    }

    return allU;
}

vector <double> gmFunc(string distributionMethod, string inputFileName, string parameter1,
→ string parameter2, bool defaultVec, bool& errorMessage) {
    ifstream inFile(inputFileName);
    string lineFile;
    vector <double> allElems;
    if (!inFile.is_open()) {
        cout << "Ошибка открытия файла " << inputFileName << endl;
        errorMessage = true;
        return allElems;
    }
    string getlineElem;
    getline(inFile, getlineElem);
    stringstream ss(getlineElem);
    while (getline(ss, getlineElem, ',')) {
        double intElem = stod(getlineElem);
        allElems.push_back(intElem);
    }
    inFile.close();

    vector <double> allU = XtoU(allElems);
    double a = stod(parameter1);
    double b = stod(parameter2);
    for (int i = 0; i < allU.size(); i++) {
        if (allU[i] == 1) {
            cout << "Одно из случайных чисел U = 1. Ошибка в вычислении ln(1 - ← U)" << endl;
            errorMessage = true;
            return allU;
        }
        allU[i] = a - (b * log(1 - allU[i]));
    }

    return allU;
}

vector <double> lnFunc(string distributionMethod, string inputFileName, string parameter1,
→ string parameter2, bool defaultVec, bool& errorMessage) {
    ifstream inFile(inputFileName);
    string lineFile;
    vector <double> allElems;

```

```

if (!inFile.is_open()) {
    cout << "Ошибка открытия файла " << inputFileName << endl;
    errorMessage = true;
    return allElems;
}

string getlineElem;
getline(inFile, getlineElem);
stringstream ss(getlineElem);
while (getline(ss, getlineElem, ',')) {
    double intElem = stod(getlineElem);
    allElems.push_back(intElem);
}
inFile.close();

if (allElems.size() % 2 != 0)
allElems.erase(allElems.end() - 1);

vector <double> allU = XtoU(allElems);
double a = stod(parameter1);
double b = stod(parameter2);
for (int i = 0; i < allU.size(); i += 2)
{
    if (allU[i] == 1) {
        cout << "Одно из случайных чисел U = 1. Ошибка в вычислении ln(1 -
        ↪ U)" << endl;
        errorMessage = true;
        return allU;
    }
    double U1 = allU[i];
    double U2 = allU[i + 1];
    double Z1 = a + (b * sqrt(-2 * log(1 - U1)) * cos(2 * M_PI * U2));
    double Z2 = a + (b * sqrt(-2 * log(1 - U1)) * sin(2 * M_PI * U2));
    double bminZ1 = b - Z1;
    if (bminZ1 > 43) {
        cout << "Разность b - Z1 больше 43. Выход за границы double в
        ↪ вычислении exp(b - Z1)" << endl;
        errorMessage = true;
        return allU;
    }
    double bminZ2 = b - Z2;
    if (bminZ2 > 43) {
        cout << "Разность b - Z2 больше 43. Выход за границы double в
        ↪ вычислении exp(b - Z2)" << endl;
        errorMessage = true;
        return allU;
    }
    allU[i] = a + exp(bminZ1);
    allU[i + 1] = a + exp(bminZ2);
}

```

```

    }

    return allU;
}

vector <double> lsFunc(string distributionMethod, string inputFileName, string parameter1,
→  string parameter2, bool defaultVec, bool& errorMessage) {
    ifstream inFile(inputFileName);
    string lineFile;
    vector <double> allElems;
    if (!inFile.is_open()) {
        cout << "Ошибка открытия файла " << inputFileName << endl;
        errorMessage = true;
        return allElems;
    }
    string getlineElem;
    getline(inFile, getlineElem);
    stringstream ss(getlineElem);
    while (getline(ss, getlineElem, ',')) {
        double intElem = stod(getlineElem);
        allElems.push_back(intElem);
    }
    inFile.close();

    vector <double> allU = XtoU(allElems);
    double a = stod(parameter1);
    double b = stod(parameter2);
    for (int i = 0; i < allU.size(); i += 2)
    {
        if (allU[i] == 1) {
            cout << "Одно из случайных чисел U = 1. Ошибка в вычислении U/(1 -
→  U). Деление на 0." << endl;
            errorMessage = true;
            return allU;
        }
        else if (allU[i] == 0) {
            cout << "Одно из случайных чисел U = 0. Ошибка в вычислении ln(U/(1
→  - U))" << endl;
            errorMessage = true;
            return allU;
        }
        allU[i] = a + (b * log(allU[i] / (1 - allU[i])));
    }

    return allU;
}

double Cnk(double n, double k) {

```

```

    double nfact = 1;
    for (int i = 1; i <= n; i++) {
        nfact = nfact * i;
    }
    double nminkfact = 1;
    for (int i = 1; i <= n - k; i++) {
        nminkfact = nminkfact * i;
    }
    double kfact = 1;
    for (int i = 1; i <= k; i++) {
        kfact = kfact * i;
    }
    double res = nfact / (nminkfact * kfact);
    return res;
}

double funcBiF(double y, double n, double p) {
    double resSum = 0;
    for (int k = 0; k <= y; k++)
    {
        resSum = resSum + (Cnk(n, k) * pow(p, k) * pow(1 - p, n - k));
    }
    return resSum;
}

vector <double> biFunc(string distributionMethod, string inputFileName, string parameter1,
→ string parameter2, bool defaultVec, bool& errorMessage) {
    ifstream inFile(inputFileName);
    string lineFile;
    vector <double> allElems;
    if (!inFile.is_open()) {
        cout << "Ошибка открытия файла " << inputFileName << endl;
        errorMessage = true;
        return allElems;
    }
    string getlineElem;
    getline(inFile, getlineElem);
    stringstream ss(getlineElem);
    while (getline(ss, getlineElem, ',')) {
        double intElem = stod(getlineElem);
        allElems.push_back(intElem);
    }
    inFile.close();

    vector <double> allU = XtoU(allElems);
    double n = stod(parameter1); //n
    double p = stod(parameter2); //p
    if (p > 1 || p < 0) {

```

```

        cout << "Ошибка. Параметр p2 не может быть > 1 или < 0, т.к. это
        ↪ вероятность." << endl;
        errorMessage = true;
        return allElems;
    }

    for (int i = 0; i < allU.size(); i++)
    {
        double y = 0;
        for (int j = 0; j <= n; j++)
        {
            if (allU[i] > funcBiF(y, n, p)) {
                y++;
                if (j == n) {
                    cout << "Ошибка. Не нашлось такого y = (0,...,n),
                    ↪ при котором U <= F(y)" << endl;
                    errorMessage = true;
                    return allElems;
                }
            }
            else
                break;
        }
        allU[i] = y;
    }

    return allU;
}

vector <double> setMethod(string distributionMethod, string inputFileName, string
↪ parameter1, string parameter2, bool defaultVec, bool& errorMessage) {
    vector <double> distributionResVec;
    if (distributionMethod == "st")
        distributionResVec = stFunc(distributionMethod, inputFileName, parameter1,
        ↪ parameter2, defaultVec, errorMessage);
    else if (distributionMethod == "tr")
        distributionResVec = trFunc(distributionMethod, inputFileName, parameter1,
        ↪ parameter2, defaultVec, errorMessage);
    else if (distributionMethod == "ex")
        distributionResVec = exFunc(distributionMethod, inputFileName, parameter1,
        ↪ parameter2, defaultVec, errorMessage);
    else if (distributionMethod == "nr")
        distributionResVec = nrFunc(distributionMethod, inputFileName, parameter1,
        ↪ parameter2, defaultVec, errorMessage);
    else if (distributionMethod == "gm")
        distributionResVec = gmFunc(distributionMethod, inputFileName, parameter1,
        ↪ parameter2, defaultVec, errorMessage);
    else if (distributionMethod == "ln")

```

```

distributionResVec = lnFunc(distributionMethod, inputFileName, parameter1,
    ↵ parameter2, defaultVec, errorMessage);
else if (distributionMethod == "ls")
distributionResVec = lsFunc(distributionMethod, inputFileName, parameter1,
    ↵ parameter2, defaultVec, errorMessage);
else if (distributionMethod == "bi")
distributionResVec = biFunc(distributionMethod, inputFileName, parameter1,
    ↵ parameter2, defaultVec, errorMessage);

return distributionResVec;
}

int main(int argc, char* argv[])
{
/*
Программа для преобразования последовательности ПСЧ
в другую последовательность ПСЧ с заданным распределением.
*/
setlocale(LC_ALL, "Rus");
string distributionMethod;
string inputFileName = "rncInput.dat";
string parameter1;
string parameter2;
bool defaultVec = true;
for (int i = 0; argv[i]; i++)
{
    string checkStr = string(argv[i]);
    if (findInStr(checkStr, 2) == "/h") {
        helpFunc();
        return 0;
    }
    if (checkStr.length() > 2) {
        string ifStr = findInStr(checkStr, 3);
        string subStr = checkStr.substr(3, checkStr.length());
        if (ifStr == "/d:") {
            distributionMethod = subStr;
        }
        if (ifStr == "/f:") {
            inputFileName = subStr;
        }
        if (ifStr == "/p1") {
            parameter1 = checkStr.substr(4, checkStr.length());
        }
        if (ifStr == "/p2") {
            parameter2 = checkStr.substr(4, checkStr.length());
        }
    }
}

```

```

}

bool errorMessage = false;
vector <double> rncRes = setMethod(distributionMethod, inputFileName, parameter1,
→ parameter2, defaultVec, errorMessage);
if (errorMessage)
return 0;
string outputFileName = distributionMethod + "Output.dat";
ofstream outFile(outputFileName);
for (int i = 0; i < rncRes.size(); i++)
{
    if (i == rncRes.size() - 1)
        outFile << (long long)rncRes[i];
    else
        outFile << (long long)rncRes[i] << ",";
}
outFile.close();
cout << "\nDistribution method: " << distributionMethod << endl;
cout << "Parameter 1: " << parameter1 << endl;
cout << "Parameter 2: " << parameter2 << endl;
cout << "Input File: " << inputFileName << endl;
cout << "Output File: " << outputFileName << endl;
return 0;
}

```