

Министерство образования и науки Российской Федерации

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

Кафедра теоретических основ
компьютерной безопасности и
криптографии

ТЕОРИЯ ПСЕВДОСЛУЧАЙНЫХ ГЕНЕРАТОРОВ

ОТЧЕТ ПО ПРАКТИЧЕСКОМУ КУРСУ

студента 4 курса 431 группы

факультета компьютерных наук и информационных технологий

Мухи Семена Андреевича

фамилия, имя, отчество

Научный руководитель

Ст. преподаватель

И.И. Слеповичев

подпись, дата

Саратов 2024

Задание 1. Генерация псевдослучайных чисел

Описание задания: создать программу, для генерации псевдослучайных величин. Входные параметры алгоритмы передаются программе в виде строки параметров, передаваемых через командную строку. Выходные значения записываются в файл, название которого указывается в строке параметров запуска программы.

Алгоритм 1. Линейный конгруэнтный метод

Описание алгоритма:

В основе линейного конгруэнтного метода лежит выбор четырех ключевых чисел:

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

Последовательность ПСЧ, получаемая по формуле:

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

называется *линейной конгруэнтной последовательностью* (ЛКП). Ключом для неё служит X_0 .

Параметры запуска программы:

/g:lc /n:10000 /i:6075;106;1283;7 /f:out.txt

Параметры алгоритма i:

Модуль, множитель, приращение, начальное значение

Исходный текст алгоритма:

```
from progress.bar import IncrementalBar
def lc(n_, parameters_):
    bar = IncrementalBar('Выполнение:', max=n_)
    res = []
    if parameters_[0]:
        m, a, c, x = int(parameters_[0]),
```

```

int(parameters_[1]), int(parameters_[2]), int(parameters_[3])

else:

    m, a, c, x = 6075, 106, 1283, 7

    for i in range(n_):

        x = (a * x + c) % m

        res.append(x)

    bar.next()

bar.finish()

return res

```

Результат работы программы:

Алгоритм 2. Аддитивный метод

Описание алгоритма:

Каждое следующее значение вычисляется по рекуррентной формуле:

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

Числа k, j – целые числа, которые называются запаздываниями, m – это модуль, n – длина вектора, который подается на вход, x_0 берется из вектора начальных значений.

Параметры запуска программы:

/g:add /n:10000 /i:128;9;49 /f:out.txt

Параметры алгоритма i:

Модуль, младший индекс, старший индекс, последовательность

начальных значений.

Исходный текст алгоритма:

```
def add(n_, parameters_):
    bar = IncrementalBar('Выполнение:', max=n_)
    res = []
    if parameters_[0]:
        xs = list(int(i) for i in parameters_) + [31, 93,
35, 97, 35, 58, 85, 32, 93, 16, 61, 78, 46, 67, 5, 66, 5, 97, 9,
83, 30, 51, 85, 84, 89, 30, 27, 53, 72, 32, 92, 13, 93, 5, 1, 97,
95, 24, 53, 28, 75, 4, 62, 4, 21, 75, 50, 6, 8, 3, 14, 62, 49,
95, 8]
    else:
        xs = [1024, 9, 49, 31, 93, 35, 97, 35, 58, 85, 32,
93, 16, 61, 78, 46, 67, 5, 66, 5, 97, 9, 83, 30, 51, 85, 84, 89,
30, 27, 53, 72, 32, 92, 13, 93, 5, 1, 97, 95, 24, 53, 28, 75, 4,
62, 4, 21, 75, 50, 6, 8, 3, 14, 62, 49, 95, 8]
    m = xs.pop(0)
    k = xs.pop(0)
    j = xs.pop(0)
    base_lan = len(xs)
    for n in range(base_lan, n_ + base_lan):
        x = (xs[n - k] + xs[n - j]) % m
        xs.append(x)
        res.append(x)
        bar.next()
    bar.finish()
    return res
```

Результат работы программы:

```
out.txt - Блокнот
Файл  Правка  Формат  Вид  Справка
38 101 19 75 12 95 34 13 73 43 70 28 30 42 18 119 97 34 73 97 81 102 74 110 4 62 39 74 66 48 126 127 10 79 66 101 78 87 123 48 5 18 82 80 35 127 54 3 55 43 119 101 27 47 94
100 68 31 22 16 102 89 127 21 3 31 51 76 77 59 32 16 43 88 44 114 0 94 35 83 51 91 47 66 102 52 47 22 121 38 92 100 36 84 32 7 22 113 8 70 108 34 89 25 107 90 16 30 86 82 123
41 2 22 2 4 27 37 92 97 100 34 109 34 77 93 5 36 67 119 118 12 35 79 96 4 15 53 84 5 7 64 38 74 100 24 117 80 118 53 44 125 62 74 68 43 30 105 48 44 38 64 96 70 47 57 14 12 13
3 102 34 93 73 13 76 120 78 48 51 111 92 41 78 55 20 91 57 75 33 32 127 15 122 86 54 72 125 9 31 1 3 5 44 35 57 72 91 65 100 116 54 67 87 41 122 58 40 75 90 88 32 32 54 70 50
9 111 37 17 80 3 24 16 4 16 60 54 82 124 106 67 37 43 85 102 90 47 25 111 68 103 13 69 95 69 63 115 39 14 64 123 124 27 5 38 48 115 41 119 77 83 14 125 86 29 98 78 8 29 86 38
80 31 124 110 0 98 66 43 14 46 68 41 58 59 18 34 45 33 80 91 75 32 111 5 73 35 101 40 5 35 95 100 1 4 106 75 99 70 93 27 22 27 55 10 26 25 60 19 68 102 58 51 120 26 123 126 62
122 46 5 41 32 96 48 104 4 19 99 64 8 40 101 41 120 126 36 125 41 103 102 81 63 109 8 122 45 90 46 91 63 50 71 13 66 102 59 53 100 89 66 90 92 122 10 42 97 94 7 71 3 124 90 58
88 94 62 123 14 123 99 56 66 126 18 36 15 47 32 21 96 45 61 70 71 53 52 18 59 25 5 110 17 79 115 16 45 88 85 127 97 95 24 47 123 33 54 38 57 64 52 73 111 83 127 116 33 71 59 2
122 109 122 16 30 105 80 104 32 118 63 116 114 78 106 16 8 121 83 93 67 69 123 10 55 125 111 20 79 16 95 98 8 91 25 126 95 83 37 30 1 114 38 74 72 11 96 41 34 123 95 32 90 102
112 21 120 3 40 25 27 59 25 114 82 44 10 18 10 61 88 51 25 107 97 44 113 27 65 96 117 85 54 8 36 67 92 127 88 53 115 97 98 42 47 37 68 4 81 25 86 117 74 31 58 60 7 121 50 113
15 118 50 73 5 11 109 127 22 75 55 109 68 78 50 65 122 12 31 60 95 43 37 58 62 102 36 39 37 78 39 3 19 95 60 53 48 50 117 109 59 119 29 64 79 127 5 116 56 74 109 79 9 84 10 11
2 47 104 68 88 124 8 20 60 110 108 19 45 30 119 33 114 5 42 114 77 38 59 112 99 7 16 63 123 40 9 8 83 3 13 116 12 62 98 98 40 52 42 10 124 66 43 110 62 42 99 18 78 84 62 51 2
73 43 23 114 10 124 11 52 93 31 100 13 35 58 8 75 87 96 6 44 104 84 79 37 31 53 54 60 115 48 22 90 69 24 22 34 10 64 12 38 74 78 40 118 60 38 12 105 19 19 121 63 104 70 34 23
16 112 94 102 120 34 90 78 117 12 10 15 127 90 22 11 76 122 21 115 73 71 102 58 13 90 13 110 66 13 59 123 67 68 126 106 67 9 127 120 61 13 90 10 28 47 40 65 13 49 125 56 112 2
2 59 56 117 29 36 127 93 73 83 61 13 121 86 109 18 116 101 67 29 54 69 95 60 7 103 115 72 127 101 89 81 126 83 5 4 32 23 90 120 119 58 114 53 98 103 37 96 52 121 117 42 42 127
70 109 45 48 125 69 107 75 76 53 68 116 85 15 92 13 100 78 112 25 44 5 5 36 116 68 74 41 108 101 23 87 126 86 86 92 54 7 56 116 32 12 68 87 79 50 37 119 57 102 121 113 7 76 11
20 117 56 49 105 21 104 122 29 34 52 1 64 8 8 106 94 125 44 109 79 43 3 61 122 20 9 119 60 11 67 119 20 104 61 88 83 18 47 70 66 26 119 69 71 53 76 21 103 86 15 47 118 48 74 5
97 120 71 60 124 101 21 9 106 105 13 121 27 30 7 33 49 37 109 118 108 37 115 61 4 31 9 27 75 70 26 30 20 103 54 69 105 32 113 91 78 90 75 83 109 64 100 79 104 47 82 18 15 105
79 126 122 78 82 123 50 104 115 32 104 32 19 27 13 69 107 110 105 110 93 74 114 119 89 3 69 60 96 78 38 16 96 26 35 104 35 83 63 100 51 105 20 32 69 66 85 43 37 2 103 14 110 2
5 25 31 98 15 0 41 32 41 55 3 20 120 1 14 103 65 76 52 20 51 51 3 101 21 88 70 69 45 53 42 63 25 29 113 37 43 23 93 65 56 94 44 27 103 116 102 85 67 61 119 75 125 118 116 15 1
123 42 46 16 70 88 63 113 58 39 59 125 87 10 52 87 8 54 108 11 33 105 60 26 71 8 98 99 123 32 67 66 14 51 63 57 43 23 53 42 115 54 76 65 36 27 14 7 67 110 96 122 81 106 115 77
2 35 23 37 81 118 90 103 40 12 3 25 17 83 19 97 116 44 28 97 107 57 15 23 61 5 18 23 52 34 47 20 10 34 112 97 82 110 55 97 32 102 48 60 120 124 73 78 68 6 9 71 97 73 114 35 53
7 58 43 81 48 49 122 43 31 65 125 106 6 76 49 125 55 101 76 50 92 73 15 8 106 81 37 63 119 113 7 57 67 51 83 99 108 32 27 118 90 108 81 51 110 102 105 32 97 117 38 124 4 30 23
0 21 38 16 56 23 97 4 89 14 57 113 113 73 105 17 51 70 50 93 105 84 123 101 13 100 119 113 7 109 79 110 45 86 81 44 36 86 103 24 90 63 80 48 42 107 39 96 19 100 84 118 64 98 2
74 121 111 1 72 105 0 23 20 9 112 16 29 83 80 17 22 18 80 13 37 68 56 71 17 63 103 94 37 67 24 122 41 60 66 73 3 63 68 116 78 31 47 44 39 1 73 65 50 24 24 30 45 111 106 73 88
116 16 20 91 71 36 16 40 15 84 49 63 111 127 22 46 96 115 3 100 76 127 85 103 54 8 120 35 14 85 38 18 85 82 16 45 126 101 39 87 17 86 56 27 3 47 43 122 104 5 102 76 118 74 83
7 81 22 58 8 12 47 81 32 44 103 27 88 83 27 44 83 51 75 57 9 17 58 6 31 104 106 46 43 48 75 104 46 17 103 98 41 53 99 126 28 24 77 32 41 82 63 115 64 35 105 99 90 49 94 110 68
108 74 35 104 62 73 36 36 57 73 51 62 89 27 96 45 126 78 85 86 64 25 69 41 79 10 14 37 84 6 10 112 74 39 123 40 98 108 104 11 50 73 75 117 72 106 65 108 69 30 19 110 93 6 51 1
98 71 42 69 127 35 7 99 94 57 30 13 124 102 31 42 121 58 86 50 99 51 53 7 70 6 94 2 23 6 76 53 16 115 38 81 88 56 81 90 75 39 106 44 117 121 34 99 78 45 110 20 113 116 28 41 7
36 36 71 15 78 120 121 64 126 120 78 9 43 10 21 96 90 60 71 107 33 52 27 25 89 102 112 84 51 71 69 48 16 120 55 121 92 103 84 36 10 10 52 61 39 16 33 53 65 46 46 123 76 117 8
32 107 72 88 6 125 36 7 42 71 66 27 88 45 24 24 95 46 103 114 55 93 1 124 34 20 120 53 72 116 117 39 4 56 75 44 92 8 24 41 69 58 105 126 123 83 104 102 4 101 37 49 86 1 80 12
```

Алгоритм 3. Пятипараметрический метод

Описание алгоритма:

Данный метод является частным случаем РСЛОС, использует характеристический многочлен из 5 членов. Генерирует последовательности -битовых двоичных чисел в соответствии с рекуррентной формулой:

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

Где X_i это биты.

Параметры q_1 , q_2 и q_3 являются показателями степени ненулевых членов характеристического полинома, а p – размер начального вектора.

Необходимо чтобы удовлетворялось условие, где

$$p > q_1 > q_2 > q_3 > 0$$

У имеет такое же значение, как и в РСЛОС.

Параметры запуска программы:

/g:5p /n:10000 /f:out.txt /i:89;20;40;69;10;12

Параметры алгоритма i:

p, q1, q2, q3, w, начальное значение

Исходный текст алгоритма:

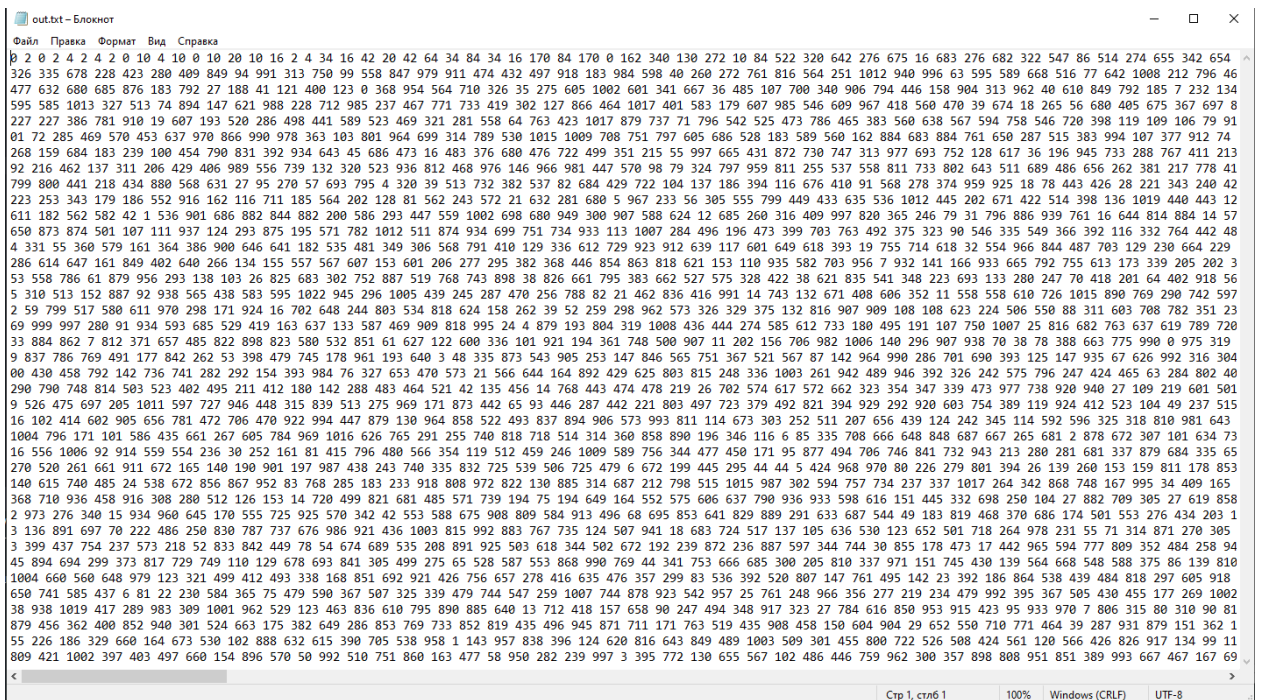
```
def p5(n_, parameters_):  
    bar = IncrementalBar('Выполнение:', max=n_)
```

```

res = []
if parameters_[0]:
    p, q1, q2, q3, w, key = int(parameters_[0]),
int(parameters_[1]), int(parameters_[2]), int(parameters_[3]), \
int(parameters_[4]),
str(parameters_[5])
else:
    p, q1, q2, q3, w, key = 89, 20, 40, 69, 10, 12
while len(key) < p:
    key = '0' + key
counter = 0
for i in range(n_):
    x_bin = ''
    for j in range(w):
        x = int(key[counter + q1]) + int(key[counter +
q2]) + int(key[counter + q3]) + int(key[counter])
        counter += 1
        x = x % 2
        key += str(x)
        x_bin += str(x)
    res.append(int(x_bin, 2))
    bar.next()
bar.finish()
return res

```

Результат работы программы:



Алгоритм 4. Регистр сдвига с обратной связью (РСЛОС)

Описание алгоритма:

Для натурального числа p и параметров a_1, a_2, \dots, a_{p-1} , которые принимают значения либо 0, либо 1, работает рекуррентная формула:

$$x_{n+p} = a_{p-1}x_{n+p-1} + a_{p-2}x_{n+p-2} + \dots + x_n \pmod{2}$$

Наименьшее положительное целое число N , такое, что $X_{n+N} = X_n$ для всех значений n называют периодом последовательности. Эту последовательность называют М-последовательностью. Период данной последовательности составляет $(2^p - 1)$.

Значения x_i и a_i лежать в диапазоне $x_1, \dots, x_p, a_1, \dots, a_p \in \{0, 1\}$.

Индексы $1 \leq j_1 < \dots < j_m \leq p$, означают, что только $a_{j_i} = 1$, остальные коэффициенты равно 0.

Параметры запуска программы:

/g:lfsrc /n:10000 /f:out.txt /i:1010110101010;541;10

Параметры алгоритма i:

Двоичное представление вектора коэффициентов, начальное значение регистра.

Исходный текст алгоритма:

```
def lfsr(flag, n_, parameters_):
    if flag:
        bar = IncrementalBar('Выполнение:', max=n_)
        taps = []
        if parameters_[0]:
            for i, j in enumerate(str(parameters_[0])[::-1]):
                if j == '1':
                    taps.append(i)
            state = int(parameters_[1])
            w = int(parameters_[2])
        else:
            for i, j in enumerate(str(101101010)[::-1]):
                if j == '1':
                    taps.append(i)
            state = 541
            w = 10
        feedback = 0
        res = []
        for i in range(n_):
            x_bin = ''
            for j in range(w):
                for tap in taps:
                    feedback ^= (state >> tap) & 1
                    state = ((state << 1) | feedback) & ((1 <<
max(taps) + 1) - 1)
            x_bin += str(state >> 1 & 1)
            res.append((int(x_bin, 2)))
            if flag:
                bar.next()
        if flag:
            bar.finish()
    return res
return res
```


Результат работы программы:

```
out.txt – Блокнот
Файл Правка Формат Вид Справка
962 250 123 126 166 482 733 698 396 763 451 889 550 963 612 139 861 21 611 645 554 739 362 800 40 92 726 731 909 567 998 341 701 362 456 620 261 931 763 183 603 650 377 116 27
1016 203 351 633 705 794 25 376 209 188 373 243 36 8 250 476 578 110 718 317 940 424 544 551 705 498 605 33 420 156 591 159 328 36 563 416 225 673 446 383 323 685 630 53 945 1
590 722 678 867 790 977 50 547 700 284 892 22 195 2 409 75 88 311 339 857 463 177 876 305 100 226 210 39 680 637 623 234 348 217 70 169 689 74 710 455 624 1002 590 233 1020 60
38 525 983 113 186 781 816 168 711 766 188 1 465 136 690 746 65 628 216 728 89 402 249 327 415 736 133 479 522 173 36 736 702 133 823 78 500 337 704 388 233 603 98 829 301 99
443 423 7 402 529 771 198 917 165 741 614 449 8 41 706 550 504 830 438 958 453 466 763 811 317 895 694 580 945 817 377 591 76 598 64 933 80 106 691 467 653 415 839 697 279 294
756 996 204 758 306 255 831 604 620 586 987 874 750 411 166 305 451 734 26 779 328 875 587 49 696 353 914 311 244 357 263 925 140 39 64 57 822 415 380 995 42 453 669 643 698 8
58 165 222 828 252 235 505 371 600 249 224 931 552 937 575 796 137 255 260 774 849 681 523 219 656 154 888 804 85 178 1015 748 746 681 48 385 941 121 680 149 43 435 41 249 892
9 582 552 890 289 888 287 271 399 788 828 347 727 305 607 440 495 196 888 460 529 491 642 716 464 709 348 429 356 5 11 602 859 497 710 1020 810 727 685 313 77 544 756 479 406
447 582 911 326 489 84 767 25 427 975 216 227 259 175 26 151 558 670 388 513 31 315 584 269 857 807 757 565 68 68 984 190 331 644 180 531 585 915 937 4 582 436 28 212 183 815
146 849 486 627 330 201 858 340 876 482 890 134 324 471 547 623 514 792 384 307 137 395 38 938 491 185 918 237 550 140 540 282 260 981 79 717 925 299 539 136 789 214 137 344 8
23 431 946 54 450 308 456 132 833 762 910 151 353 742 21 88 991 791 512 186 145 86 349 264 206 539 91 11 178 287 168 947 988 16 699 961 277 644 604 87 784 742 905 830 554 216
967 342 29 1005 635 196 55 436 896 946 322 224 408 882 660 732 716 824 129 5 216 324 831 103 822 887 824 698 351 485 423 751 982 840 630 230 175 201 905 586 776 116 650 13 342
27 482 321 476 121 308 1011 478 636 537 606 806 287 999 971 589 585 379 493 349 819 404 806 184 475 771 353 425 109 457 390 215 44 242 294 926 556 672 1011 657 516 904 7 230 8
916 211 81 28 7 937 843 574 37 581 788 667 871 543 541 447 174 459 31 156 116 453 117 199 995 529 159 928 608 874 213 193 411 466 19 367 100 522 662 382 989 605 341 134 48 245
24 13 958 1022 136 966 968 237 206 712 837 111 292 239 35 929 945 994 615 555 474 934 203 951 61 920 623 57 578 189 464 345 570 88 683 565 684 512 641 459 363 446 216 895 613
97 916 0 847 120 401 601 960 556 72 439 968 881 936 829 138 607 899 181 505 923 28 416 405 899 274 965 851 816 576 131 935 457 33 747 228 990 710 648 520 635 23 809 464 534 57
371 99 419 477 832 1016 24 577 541 855 746 146 362 188 846 425 281 235 298 621 572 367 272 808 570 964 461 960 355 48 38 401 177 388 885 317 407 242 797 708 785 579 547 288 63
5 277 459 548 454 329 294 421 886 925 784 833 437 1014 262 824 294 569 16 616 223 369 786 940 220 770 651 123 994 960 23 274 138 811 673 25 835 395 385 406 291 917 118 507 514
4 949 747 939 934 855 209 456 87 95 670 536 359 490 756 120 938 771 765 719 408 518 950 624 118 296 284 51 110 338 603 601 615 16 128 667 40 615 908 998 878 999 87 299 956 692
87 726 224 727 778 773 133 268 276 201 434 784 565 151 858 956 296 187 527 166 638 443 847 579 203 868 803 1020 1017 457 713 175 445 683 742 434 612 791 59 480 428 181 141 697
3 574 874 573 645 194 167 51 597 8 870 698 951 929 254 922 1014 114 538 394 131 512 1013 233 455 772 712 738 595 492 963 963 695 917 825 387 915 526 568 654 664 1020 450 147 1
507 933 619 304 910 304 861 558 825 952 713 819 731 862 297 898 195 845 481 474 513 759 895 785 120 889 29 665 857 104 653 932 541 900 500 246 252 332 965 443 372 793 502 903
365 90 125 518 869 366 18 190 236 184 124 319 425 645 909 223 418 12 968 330 754 512 105 911 50 203 248 69 521 54 1017 110 245 103 657 331 1008 406 703 243 387 564 50 752 418
83 958 278 716 671 701 205 1012 676 553 67 1005 147 640 366 193 928 136 302 396 436 443 744 127 3 72 195 746 989 274 301 279 617 821 163 157 421 333 711 557 930 101 71 376 569
882 124 152 917 589 161 831 180 40 787 686 842 468 503 458 285 778 674 697 452 568 809 164 820 750 883 738 104 182 766 800 871 36 839 130 77 27 942 226 373 539 608 337 399 508
8 685 77 367 652 78 975 11 1021 359 313 490 28 572 755 630 733 501 500 874 922 185 10 907 979 835 44 957 350 527 117 352 479 729 947 64 886 846 14 805 35 518 397 810 331 459 2
6 1008 522 473 262 588 4 661 682 120 322 327 932 954 440 316 986 796 90 993 352 656 673 546 537 182 354 70 658 1003 375 549 23 449 916 847 823 489 968 409 492 612 511 639 185
563 371 1023 709 40 143 456 703 539 967 877 327 720 664 276 902 458 641 108 855 374 1012 159 883 382 782 323 305 272 448 126 669 184 992 601 92 330 445 632 504 470 1010 743 17
814 990 865 180 704 343 1015 575 500 717 422 113 806 107 709 871 247 89 230 475 491 805 240 800 489 700 187 320 222 1007 994 143 111 131 723 235 141 81 756 579 752 574 542 799
8 540 189 824 797 867 301 651 271 704 876 685 770 279 797 535 15 551 949 208 753 667 948 257 633 41 350 320 13 241 902 281 415 8 705 134 895 141 798 652 978 169 510 50 855 926
354 306 944 839 973 53 394 503 802 857 595 983 665 766 596 581 136 509 658 464 45 792 244 17 37 817 566 567 477 15 896 393 24 477 379 674 293 674 973 230 660 403 692 681 728 9
2 259 124 772 243 952 878 271 531 114 713 660 231 918 517 98 469 873 314 574 953 291 737 340 343 184 583 101 148 614 605 878 476 269 22 863 868 108 900 616 912 265 643 501 796
7 733 617 658 671 85 649 685 977 521 889 897 511 684 935 189 259 583 606 462 859 702 621 371 279 129 369 506 488 389 631 683 833 910 684 59 987 246 392 110 873 769 868 644
4 663 583 273 438 714 231 382 65 315 160 841 512 594 725 271 40 296 1012 631 311 39 635 355 523 380 172 82 84 196 323 150 812 264 850 381 430 964 642 952 242 617 998 957 249 5
0 234 871 387 379 74 865 1019 696 198 430 511 984 645 17 953 87 963 504 1005 680 986 83 34 624 825 336 141 618 942 894 531 1006 431 865 808 422 162 56 15 851 663 124 75 139 55
62 676 910 639 805 959 352 120 229 891 876 150 600 42 1022 967 958 601 692 782 228 781 472 748 926 907 156 827 445 484 670 35 61 215 535 424 27 893 1020 273 909 912 474 413 40
13 710 179 850 838 244 761 609 616 835 535 743 99 748 421 721 417 984 109 597 736 290 995 706 897 964 1014 666 94 211 502 544 207 133 171 808 1 670 240 803 179 897 88 144 879
2 857 1012 491 593 466 436 851 172 294 374 104 1017 646 689 318 996 363 202 506 979 223 842 584 657 63 722 314 5 739 30 514 132 742 198 838 955 641 1008 49 131 59 687 468 292
4 668 361 872 235 694 267 969 672 800 399 608 542 503 109 801 962 398 345 210 540 1010 832 652 314 749 167 327 887 164 476 170 554 919 72 908 658 588 843 749 827 545 642 875 1
```

Алгоритм 5. Нелинейная комбинация РСЛОС. Генератор Геффа

Описание алгоритма:

Генератор Геффа является примером нелинейной комбинацией РСЛОС.

В этом генераторе используются три РСЛОС, объединённые нелинейным образом. Длины этих регистров L_1, L_2, L_3 - попарно простые числа.

Нелинейная функция генератора:

$$f(x_1, x_2, x_3) = x_1 x_2 \oplus x_2 x_3 \oplus x_3$$

Параметры запуска программы:

/g:nfsr /n:10000 /f:out.txt /i:1010110101010;10101101010;10101101010;541;993;117;10

Параметры алгоритма i:

Параметры – двоичное представление векторов коэффициентов для R1, R2, R3, w, x1, x2, x3. w – длина слова, x1, x2, x3 – десятичное представление начальных состояний регистров R1, R2, R3.

Исходный текст алгоритма:

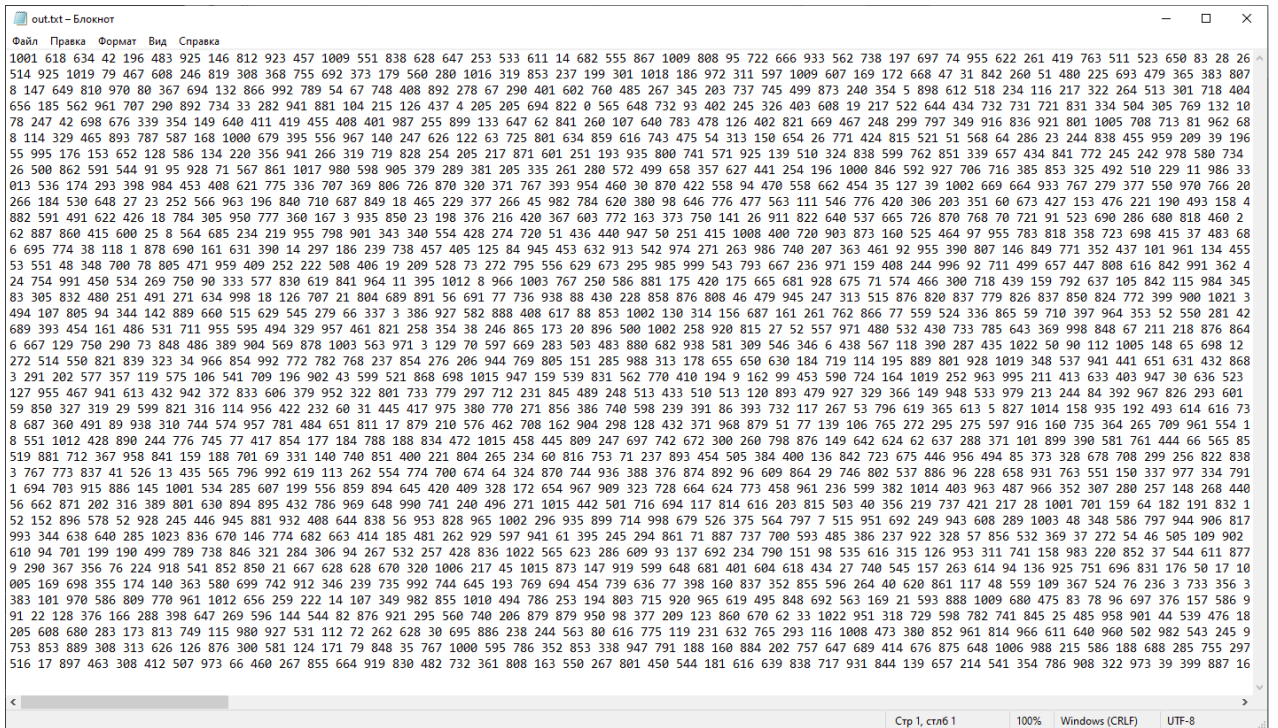
```
def nfsr(n_, parameters_):  
    bar = IncrementalBar('Выполнение:', max=n_)
```

```

result = []
if parameters_[0]:
    w = parameters_[6]
    x1 = lfsr(False, n_, [parameters_[0],
parameters_[3], w])
    x2 = lfsr(False, n_, [parameters_[1],
parameters_[4], w])
    x3 = lfsr(False, n_, [parameters_[2],
parameters_[5], w])
else:
    w = 10
    x1 = lfsr(n_, [1010110101010, 541, w])
    x2 = lfsr(n_, [1010110101010, 993, w])
    x3 = lfsr(n_, [1010110101010, 117, w])
for i in range(0, n_):
    x1Element = x1[i]
    x2Element = x2[i]
    x3Element = x3[i]
    resultTemp = (x1Element & x2Element) ^ (x2Element &
x3Element) ^ x3Element
    result.append(resultTemp)
    bar.next()
bar.finish()
return result

```

Результат работы программы:



Алгоритм 6. Вихрь Мерсенна

Описание алгоритма:

Метод Вихрь Мерсенна позволяет генерировать последовательность двоичных псевдослучайных целых w -битных чисел в соответствии с рекуррентной формулой:

$$x_{n+p} = x_{n+q} \oplus \left(X_{n+1}^l \right) A \quad (n = 0, 1, 2, \dots),$$

где p , q , r – целые константы;

p – степень рекуррентности, $1 \leq q \leq p$;

X_n – w -битное двоичное целое число;

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

A – матрица размера $w \times w$ состоящая из нулей и единиц, определенная

посредством a .

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

Параметры задаются изначально:

w – размер слова (разрядность значений, которыми оперирует алгоритм)

$r: r \leq w$ – позиция разделения

$p, q: 0 < q \leq p$ – два положительных числа

$a, b, c: 0 \leq a, b, c < 2^w$ – w -разрядные неотрицательные числа

$u, s, t, l: 0 \leq u, s, t, l \leq w$ – коэффициенты

x_0, \dots, x_{p-1} – начальные значения вектора

Параметры запуска программы:

/g:mt /n:10000 /f:out.txt /i:624

Параметры алгоритма i:

Модуль, начальное значение x

Исходный текст алгоритма:

```
def mt(n_, parameters_):
    bar = IncrementalBar('Выполнение:', max=n_)
    w = 32
    r = 31
    q = 397
    a = 2567483615
    u = 11
    s = 7
    t = 15
    l = 18
    b = 2636928640
    c = 4022730752
    if parameters_[0]:
        p = int(parameters_.pop(0))
    else:
        p = 624
```

```
if parameters_[0]:
    x = parameters_
else:
    x = [5, 28, 14, 27, 18, 17, 23, 1, 24, 17, 5, 11, 1,
1, 19, 13, 21, 7, 2, 11, 8, 23, 21, 15, 24, 4, 7, 11, 29, 7, 24,
15, 13, 9, 26, 3, 12, 9, 8, 17, 2, 2, 28, 6, 30, 14, 29, 21, 13,
1, 27, 9, 18, 26, 8, 14, 22, 15, 9, 7, 21, 10, 12, 6, 23, 17,
13, 18, 3, 29, 29, 17, 5, 14, 18, 18, 17, 12, 5, 3, 18, 26, 29,
29, 7, 17, 1, 23, 16, 9, 26, 28, 4, 21, 6, 30, 29, 15, 14, 26,
24, 30, 23, 26, 22, 22, 26, 9, 2, 16, 11, 16, 30, 3, 7, 30, 8,
11, 9, 10, 25, 12, 1, 22, 11, 5, 22, 12, 24, 18, 17, 6, 10, 15,
21, 24, 26, 12, 13, 4, 19, 26, 26, 22, 15, 10, 1, 18, 25, 28, 1,
24, 18, 27, 3, 5, 15, 27, 21, 17, 5, 29, 16, 28, 30, 10, 26, 6,
22, 6, 4, 4, 8, 5, 2, 4, 17, 22, 5, 12, 15, 11, 8, 7, 5, 5, 8,
18, 23, 28, 19, 2, 18, 6, 2, 3, 9, 17, 9, 4, 29, 6, 29, 17, 25,
11, 18, 28, 12, 6, 13, 8, 14, 14, 7, 13, 9, 22, 28, 20, 30, 3,
8, 1, 28, 10, 28, 7, 12, 26, 14, 27, 18, 30, 7, 18, 2, 30, 13,
12, 11, 17, 9, 24, 23, 10, 18, 4, 15, 29, 3, 19, 15, 24, 13, 15,
7, 12, 3, 7, 21, 17, 24, 3, 14, 22, 9, 29, 2, 7, 27, 29, 18, 26,
4, 12, 6, 25, 21, 8, 20, 11, 10, 23, 8, 4, 26, 28, 12, 19, 11,
13, 1, 3, 22, 12, 16, 11, 6, 26, 28, 17, 25, 29, 5, 12, 27, 25,
11, 7, 13, 5, 27, 12, 19, 25, 11, 5, 3, 5, 9, 26, 28, 25, 18,
18, 22, 16, 17, 29, 2, 20, 2, 7, 2, 26, 29, 6, 6, 23, 8, 20, 6,
26, 24, 28, 22, 15, 7, 28, 26, 7, 24, 21, 28, 16, 27, 8, 2, 3,
19, 23, 6, 20, 19, 27, 16, 16, 1, 20, 10, 8, 8, 8, 28, 21, 8,
11, 4, 13, 29, 29, 8, 24, 22, 3, 2, 26, 13, 19, 8, 17, 25, 6, 2,
7, 4, 20, 24, 26, 2, 23, 9, 15, 22, 19, 20, 24, 29, 2, 29, 24,
3, 28, 30, 2, 22, 28, 21, 28, 9, 12, 30, 18, 13, 2, 9, 17, 20,
10, 24, 30, 20, 23, 6, 30, 21, 8, 26, 13, 30, 9, 30, 1, 14, 19,
16, 6, 18, 9, 15, 1, 27, 4, 12, 4, 26, 6, 24, 19, 24, 4, 15, 6,
13, 10, 24, 2, 29, 5, 12, 24, 14, 24, 11, 1, 23, 24, 12, 2, 2,
18, 27, 30, 11, 26, 28, 20, 20, 8, 11, 23, 4, 26, 19, 17, 21,
11, 29, 11, 30, 2, 9, 12, 17, 18, 18, 13, 14, 12, 19, 20, 7, 15,
2, 17, 15, 26, 12, 24, 22, 3, 4, 22, 16, 9, 12, 16, 13, 30, 14,
24, 1, 10, 21, 16, 6, 1, 30, 27, 19, 25, 27, 7, 12, 17, 24, 29,
12, 20, 4, 21, 12, 16, 13, 21, 23, 29, 2, 29, 21, 12, 13, 23,
```

```

12, 22, 16, 12, 19, 22, 6, 20, 11, 28, 16, 7, 26, 14, 17, 17, 4,
22, 29, 6, 27, 14, 16, 28, 18, 11, 25, 2, 13, 27, 14, 23, 27,
14, 30, 21, 6, 6, 4, 12, 15, 17, 27, 3, 6, 5, 2, 19, 9, 12, 24,
20, 11, 21, 13, 8, 26, 16, 18, 1]

    res = []
    value1 = ''
    value2 = ''
    for i in range(w - r):
        value1 += '1'
        value2 += '0'
    for i in range(r):
        value1 += '0'
        value2 += '1'
    value1Int = int(value1, 2)
    value2Int = int(value2, 2)
    n = int(n_)
    for i in range(n + 1500):
        t12 = int(x[i]) & value1Int
        t13 = int((x[i + 1])) & value2Int
        Y = t12 | t13
        if (Y % 2 != 0):
            valuex = (int(x[i + q]) % 2 ** w) ^ (Y >> 1) ^ a
        else:
            valuex = (int(x[i + q]) % 2 ** w) ^ (Y >> 1) ^ 0
        Y = valuex
        Y = (Y ^ (Y >> u))
        Y = Y ^ ((Y << s) & b)
        Y = Y ^ ((Y << t) & c)
        Z = (Y ^ (Y >> 1))
        x.append(valuex)
        res.append(Z % p)
    bar.next()
bar.finish()
return res

```

Результат работы программы:

```
out.txt – Блокнот
Файл Правка Формат Вид Справка
523 222 361 245 323 127 425 181 553 290 402 246 335 542 223 225 420 532 452 267 274 465 175 251 611 523 441 50 442 281 481 319 36 514 425 553 77 509 622 100 57 102 489 125 236
18 595 144 212 440 238 564 553 621 547 467 400 526 412 60 48 430 65 281 414 495 239 79 496 192 374 340 545 333 270 564 141 428 56 356 402 470 384 560 154 528 611 341 426 329 7
587 76 342 43 410 410 149 486 76 136 66 357 400 210 567 597 39 219 597 376 84 91 463 263 353 100 401 438 183 368 167 214 560 228 200 176 143 593 449 397 313 347 425 492 457 52
418 560 181 50 543 15 179 409 32 28 98 199 17 509 316 441 326 193 453 547 571 288 187 478 191 73 549 180 124 237 524 255 187 385 612 561 436 385 564 580 64 527 41 254 96 188 5
94 265 619 614 609 345 477 492 398 124 319 611 220 119 367 361 86 54 553 66 285 381 155 249 393 581 479 318 16 430 325 295 192 464 122 536 398 16 116 570 201 138 144 72 104 60
264 183 589 492 547 433 423 112 465 528 342 288 490 88 542 247 545 167 372 409 60 539 153 543 338 309 394 21 317 135 607 293 240 301 50 201 285 241 130 294 237 535 268 122 170
29 149 184 155 339 333 504 510 480 410 112 97 271 535 414 125 411 372 321 2 74 429 138 362 171 16 208 361 407 542 433 113 563 461 458 418 326 497 521 93 338 223 9 1 10 537 120
357 490 342 208 614 41 361 403 310 205 342 163 502 329 465 62 591 590 444 208 428 175 234 253 563 222 406 618 591 588 389 395 368 491 384 264 276 101 191 295 589 123 303 491 7
4 252 609 239 419 458 283 595 429 52 472 357 492 615 299 473 364 386 368 307 574 162 395 486 67 542 487 419 546 325 443 594 125 119 476 19 260 440 220 53 13 267 7 458 214 198
492 225 49 51 346 436 614 600 44 412 213 76 75 321 13 63 297 519 472 418 491 116 218 29 224 200 357 490 32 117 221 38 102 236 114 520 242 505 18 21 497 573 396 65 252 308 512
39 155 223 265 439 542 295 252 79 509 581 378 202 453 549 567 93 491 416 109 80 83 34 605 284 481 227 322 310 429 218 370 547 53 410 349 379 514 550 222 471 147 391 448 148 6
0 92 345 434 212 225 406 495 63 264 582 92 576 81 108 204 375 29 545 467 261 243 110 405 193 548 309 376 212 36 553 151 48 362 502 417 79 348 348 551 591 552 608 136 200 335 3
483 570 609 461 287 506 47 395 584 524 464 184 58 32 218 87 332 372 563 363 184 9 530 366 246 130 593 194 563 309 27 381 531 377 370 580 503 469 549 456 488 191 163 197 623 58
209 77 27 207 55 124 146 378 341 262 417 72 37 567 319 355 102 237 224 402 177 527 187 224 540 551 489 263 1 459 512 140 333 393 67 377 238 4 141 501 78 383 572 605 173 310 38
279 477 122 398 477 37 260 453 527 151 21 442 491 316 232 609 294 177 512 557 249 244 518 85 49 430 94 430 227 526 185 337 68 439 412 253 382 268 609 323 49 545 218 488 569 14
177 286 283 614 610 251 328 112 127 448 120 294 446 528 466 565 249 313 546 164 281 207 145 46 266 37 424 65 366 86 591 154 293 165 80 102 284 401 119 604 421 216 318 545 34 5
252 487 543 133 337 448 537 220 479 68 443 358 603 399 137 218 368 616 272 110 416 423 240 486 6 410 322 196 534 552 179 154 376 394 458 347 293 208 76 261 469 589 169 468 530
03 12 257 619 360 14 336 498 63 622 0 231 19 383 207 107 32 118 58 200 106 474 270 74 295 453 293 206 598 114 106 333 352 506 192 213 487 438 359 353 354 333 84 166 130 466 49
373 436 130 317 52 600 590 568 373 251 252 293 621 325 449 473 341 328 384 564 424 9 230 228 25 556 74 433 448 368 430 52 49 217 408 615 612 304 522 493 617 46 593 553 221 72
27 253 239 558 129 235 224 568 396 117 601 200 332 508 406 214 304 193 321 341 419 224 35 373 62 368 512 182 103 194 358 157 173 509 379 602 437 159 443 448 295 413 559 377 36
62 77 540 23 491 457 613 270 191 395 412 587 101 53 160 140 359 437 563 332 138 328 486 244 295 405 269 102 258 199 87 485 379 122 17 356 302 350 337 34 220 581 496 379 324 57
68 209 463 479 203 306 144 536 594 584 600 212 353 522 46 86 593 161 334 557 361 449 90 199 412 134 490 189 251 472 99 19 47 214 431 552 355 205 526 262 236 115 398 132 579 33
326 314 219 601 138 531 242 524 442 199 542 602 578 67 285 126 182 590 585 570 55 473 532 509 531 588 541 592 390 17 505 554 353 527 321 401 64 427 522 508 534 591 578 512 331
53 77 590 16 474 600 492 282 289 72 485 612 620 82 531 131 562 456 373 264 382 186 210 417 618 95 140 453 606 7 48 401 151 257 173 327 423 613 321 581 377 139 422 213 11 275 3
463 203 264 241 19 127 88 526 429 542 431 224 80 271 546 194 286 584 402 336 261 54 105 445 210 287 74 439 240 232 249 324 225 246 403 37 473 344 500 129 134 4 320 118 64 176
51 351 401 81 84 266 97 160 14 270 209 372 567 469 437 337 342 127 533 150 484 208 95 450 532 554 3 319 550 563 547 580 511 238 578 68 95 272 612 59 433 273 0 340 153 565 619
414 415 376 86 42 364 204 181 407 281 286 85 35 407 290 22 19 342 327 581 394 41 236 194 113 410 358 541 176 433 563 582 249 35 43 329 169 185 252 501 501 256 151 119 221 207
400 134 106 180 556 601 185 94 583 339 426 148 286 10 215 197 28 551 289 304 333 565 327 32 63 459 400 115 38 129 50 50 411 305 316 268 183 290 616 121 285 75 248 379 222 445
429 398 117 230 187 299 131 511 548 179 177 606 535 533 337 12 102 48 131 389 140 191 539 407 532 168 532 459 212 68 527 417 588 157 611 39 96 527 514 203 266 298 462 307 590
163 321 154 392 1 104 25 177 563 252 47 448 66 309 413 158 135 520 298 614 201 270 283 512 31 86 292 90 242 195 509 562 441 145 470 2 104 582 130 551 346 164 68 599 125 92 268
189 575 300 296 180 275 53 501 268 545 336 506 523 436 526 432 383 136 505 486 614 598 120 121 266 148 554 53 265 409 397 31 88 419 130 267 315 245 59 567 52 327 434 287 254 3
574 614 279 103 604 227 302 202 591 364 400 338 543 569 493 544 105 609 546 558 290 41 620 161 82 554 462 268 149 222 451 531 424 165 18 463 291 616 562 132 93 390 168 241 434
162 190 601 74 555 15 620 464 143 442 202 403 78 57 20 581 503 151 465 520 151 450 436 596 315 289 70 609 332 163 104 143 391 554 505 263 499 457 355 407 93 551 33 107 350 251
295 606 208 143 485 156 516 239 138 353 444 473 452 358 508 377 331 408 255 613 112 293 477 331 395 478 600 402 51 55 26 171 574 525 489 438 202 182 214 547 233 620 94 69 17 4
343 526 237 244 534 303 501 421 204 491 21 487 36 430 199 356 107 324 591 329 405 298 512 92 125 383 259 269 594 463 31 453 191 344 223 496 414 603 95 464 393 318 96 106 57 26
148 488 443 435 116 368 487 120 571 102 53 307 532 99 544 147 503 303 575 609 586 496 283 538 234 46 37 21 427 58 545 245 508 551 434 607 124 109 523 453 5 598 564 315 86 68 3
5 160 527 404 63 207 257 422 62 437 14 177 164 6 354 368 296 150 586 350 481 391 261 146 48 572 570 234 402 584 611 158 304 154 442 588 67 199 3 264 32 116 68 478 485 68 533 1
67 431 166 395 613 246 619 148 183 464 475 108 314 358 93 34 365 469 344 283 535 143 325 176 328 396 311 477 176 581 110 147 345 521 506 323 598 195 480 357 545 594 581 521 56
```

Алгоритм 7. RC4

Описание алгоритма:

На вход:

n – количество необходимых сгенерированных чисел

w – количество бит, используемых для генерации числа

K – массив ключа длины 256, которые состоит из чисел от 0 до 255, которые перемешаны любым способом.

1. Инициализация $S_i, i = 0, 1, ..., 255$.

Начальное значение состояния заполняется на основе ключа $K_0, ..., K_{255}$

a) $for i = 0 \ 255 : S_i = i;$

b) $j = 0;$

c) $for i = 0 \ to \ 255: j = (j + S_i + K_i) \ mod \ 256; \ Swap(S_i, S_j)$

2. $i = 0, j = 0$.

3. Итерация алгоритма:

a) $i = (i + 1) \ mod \ 256;$

$$b) j = (j + S_i) \bmod 256;$$

$$c) \text{Swap}(S_i, S_j);$$

$$d) t = (S_i + S_j) \bmod 256;$$

$$e) K = S_t;$$

Параметры запуска программы:

/g:rc4 /n:10000 /f:out.txt

/i:73;25;169;67;200;69;83;93;19;100;141;85;207;66;71;236;194;239;167;
32;101;135;213;35;89;112;188;178;82;33;206;54;249;51;255;102;164;155;133;46;
16;231;152;42;122;15;41;14;208;244;230;6;8;245;217;124;227;185;184;248;37;59
;31;191;120;111;26;253;140;63;125;242;3;136;27;36;186;95;220;94;243;49;70;15
0;79;118;117;176;172;247;24;65;241;238;174;55;114;21;2;129;162;17;210;254;22
;60;62;251;91;215;0;109;223;156;97;11;127;123;130;250;192;1;138;52;113;160;1
81;229;105;47;44;40;180;116;168;153;154;201;72;234;128;224;5;161;197;134;13
2;190;177;29;56;12;77;50;58;74;131;146;246;68;166;96;216;219;212;13;115;211;
157;144;203;20;232;9;45;34;23;151;195;237;196;80;57;7;205;43;182;193;106;87;
104;119;38;218;148;48;187;221;226;159;145;202;110;228;173;209;10;108;75;84;
139;53;61;18;103;183;98;179;165;81;126;137;171;170;252;147;78;214;163;158;2
33;149;142;175;121;76;90;4;92;199;30;107;88;99;189;222;225;143;235;39;240;20
4;28;86;198

Параметры алгоритма i:

256 начальных значений

Исходный текст алгоритма:

```
def rc4(n_, parameters_):
    res = []
    bar = IncrementalBar('Выполнение:', max=n_)
    length = 256
    if parameters_[0]:
        secretKey = parameters_
    else:
```



```

        secretKey = ['1', '2', '3', '4', '5', '6', '7',
'31', '48', '178', '46', '91', '30', '15', '132', '154',
'108', '222', '245', '211', '103']

sBox = [0] * length
key = [0] * length
for i in range(length):
    sBox[i] = i
for i in range(length):
    key[i] = int(secretKey[i % len(secretKey)])
j = 0
for i in range(length):
    j = (j + sBox[i] + key[i]) % length
    sBox[i], sBox[j] = sBox[j], sBox[i]
j = 0
for i in range(n_, 0, -1):
    i = (i + 1) % length
    j = (j + sBox[i]) % length
    sBox[i], sBox[j] = sBox[j], sBox[i]
    t = (sBox[i] + sBox[j]) % length
    res.append(sBox[t])
    bar.next()
bar.finish()
return res

```

Результат работы программы:

out.txt – Блокнот

Файл Правка Формат Вид Справка

```
186 93 145 165 218 84 12 188 228 125 156 165 21 179 167 73 34 90 75 74 58 109 130 141 30 243 121 91 98 1 219 205 47 61 172 52 110 23 193 158 100 188 216 107 236 16 61 247 102
6 228 86 169 228 117 108 62 205 105 210 115 124 187 4 58 176 77 155 0 195 36 135 1 89 30 105 100 3 26 248 27 38 93 28 231 107 209 62 90 84 171 100 41 44 128 4 217 246 231 31 1
60 21 16 35 3 191 189 200 165 238 37 25 210 154 241 63 186 218 19 154 68 182 207 71 180 137 81 36 255 99 40 235 207 70 33 81 170 209 42 190 196 124 188 235 115 71 123 118 60 6
220 74 174 65 159 198 164 83 222 118 4 141 63 41 224 224 29 78 86 159 225 28 60 213 32 94 99 165 213 72 244 74 10 5 230 214 18 196 24 32 255 74 124 110 30 77 229 136 98 237 23
197 153 75 215 188 93 75 18 18 13 92 42 106 216 8 120 101 130 20 39 204 219 63 16 193 191 121 168 212 217 50 59 1 59 197 113 227 86 182 162 165 123 196 177 227 23 23 242 76 17
0 86 28 223 221 183 42 133 70 174 212 98 155 92 44 35 242 240 212 137 41 176 116 206 68 11 49 245 114 91 213 109 230 156 158 146 12 151 250 125 231 224 166 149 146 165 210 168
215 29 134 132 115 181 247 54 169 124 169 97 164 254 118 40 155 194 222 186 160 249 127 242 100 165 51 157 23 209 168 3 230 215 5 21 152 125 101 80 27 47 14 234 37 186 43 74 7
33 7 85 195 254 187 8 140 115 122 112 189 250 22 243 7 247 239 65 150 21 204 164 101 179 207 205 239 173 229 40 171 171 231 142 100 69 228 168 191 228 106 39 171 52 76 30 234
16 60 133 94 231 118 146 65 200 28 106 141 203 85 207 29 43 29 185 223 34 52 145 60 20 149 2 14 206 143 181 107 253 152 218 212 54 244 193 74 17 227 59 190 114 247 73 70 115 4
248 40 77 195 39 143 108 229 15 203 31 2 135 2 108 43 29 51 105 106 222 138 9 211 0 29 44 233 6 88 80 41 82 86 218 39 30 160 88 157 232 117 214 23 129 176 153 68 138 60 209 12
77 81 165 222 107 173 66 36 15 18 239 48 83 21 238 22 88 65 29 91 120 204 212 140 58 77 11 43 186 171 201 38 61 129 153 218 242 233 185 69 150 99 1 157 19 4 120 228 57 45 208
64 20 45 110 20 132 130 115 42 247 107 184 16 21 78 127 171 93 181 218 21 206 75 113 254 38 148 180 254 7 98 23 89 68 47 204 45 243 188 73 172 121 136 74 244 94 75 243 117 159
195 223 27 29 14 156 135 243 175 34 254 254 152 162 114 175 19 19 151 55 105 67 193 25 140 50 216 124 194 197 28 99 104 143 218 125 62 5 125 236 107 67 1 22 162 104 245 108 12
0 209 117 251 240 192 81 13 43 226 7 94 130 143 191 159 254 54 244 39 26 82 131 130 100 98 198 84 81 76 36 136 54 151 153 24 221 48 1 222 35 95 95 211 11 125 23 252 189 247 97
177 19 191 205 172 142 157 65 121 121 167 115 113 173 167 234 78 77 64 250 36 118 18 145 202 66 62 45 158 179 146 98 110 87 129 144 60 34 120 107 16 41 82 58 251 106 144 32 32
151 2 122 98 132 136 65 155 130 108 217 191 243 66 213 34 212 174 166 214 56 150 44 63 197 134 120 176 185 171 137 128 74 42 55 104 130 207 238 110 119 31 114 82 150 148 135 1
58 50 2 102 17 211 39 29 20 52 94 129 201 101 5 116 216 204 246 245 114 3 249 148 146 174 100 237 227 104 155 147 59 145 3 124 8 166 121 238 63 21 192 6 150 242 173 222 78 90
3 86 188 1 41 221 83 33 206 106 204 82 83 34 17 41 177 243 147 58 69 154 45 235 70 106 11 144 157 41 12 181 205 175 76 186 214 43 172 135 159 195 166 232 77 118 115 0 58 10 17
6 59 238 186 166 124 117 0 200 143 34 214 136 26 101 59 71 223 169 116 210 145 81 22 160 47 103 159 217 236 87 91 155 178 201 245 62 133 2 169 126 223 216 163 230 81 42 21 60
62 162 137 232 199 220 196 10 208 178 24 135 27 138 8 107 6 206 204 235 69 209 196 14 82 232 155 33 238 27 220 133 54 45 191 116 198 224 33 153 46 169 7 50 229 5 47 184 75 29
44 202 76 89 97 89 55 182 14 78 205 24 133 10 219 209 96 206 118 78 211 243 217 200 143 56 18 203 152 40 23 225 149 204 206 26 106 220 133 33 104 59 103 79 188 64 252 24 106 8
101 144 72 72 127 50 133 251 177 64 162 4 88 112 5 40 164 63 250 106 144 5 247 12 191 88 106 149 106 13 51 36 207 201 45 116 226 234 94 7 242 100 185 215 129 110 19 245 108 18
58 38 88 104 49 244 85 86 10 27 225 41 222 62 14 181 216 149 223 103 27 145 17 67 194 75 42 176 107 9 53 51 60 53 151 178 57 17 36 61 17 109 126 37 107 39 98 34 148 197 107 25
199 133 130 140 54 92 29 156 178 159 68 136 171 212 64 82 163 155 156 16 55 161 34 47 125 27 2 113 63 241 237 65 163 147 29 115 149 182 116 24 218 138 32 51 94 40 215 191 53 1
30 100 95 150 83 188 227 125 199 239 213 52 203 196 179 246 204 190 99 196 162 48 117 34 217 250 36 172 237 123 200 5 11 106 234 153 122 224 82 160 46 220 26 127 143 27 155 12
128 240 235 175 195 29 36 98 173 60 144 97 210 80 163 137 157 212 183 33 220 193 107 117 18 239 156 172 16 216 198 57 41 213 213 254 212 163 19 248 97 206 106 252 20 70 172 20
142 127 43 172 128 159 104 12 176 158 89 104 217 243 29 68 233 225 176 4 79 63 136 178 10 187 148 214 210 154 161 154 90 225 185 84 170 81 119 234 136 12 126 112 114 231 124 1
6 183 242 197 18 183 60 164 19 166 194 113 215 210 197 94 128 248 178 91 160 162 182 179 218 18 102 29 22 120 150 220 242 98 182 90 215 100 206 160 109 18 164 223 42 10 197 13
253 212 150 147 151 222 133 189 89 70 245 62 14 210 207 189 187 43 65 39 255 97 190 22 138 226 153 75 73 143 110 68 109 148 31 73 60 41 248 240 30 254 16 25 8 51 153 131 198 0
6 86 185 17 28 9 32 244 74 184 106 172 97 183 88 35 204 225 84 248 74 96 235 120 38 98 241 91 221 45 210 71 250 20 12 122 210 210 9 40 111 119 161 129 150 211 182 87 133 144 8
251 108 205 172 19 200 11 202 135 204 7 165 17 69 132 226 12 123 114 95 47 212 245 3 150 80 251 215 195 104 35 208 128 61 243 51 242 147 56 168 44 175 229 86 13 153 206 161 17
72 137 209 235 200 157 235 198 111 120 95 111 159 121 118 153 88 151 126 171 170 183 135 239 187 62 111 192 154 86 53 215 93 119 137 248 183 43 143 17 147 214 197 20 8 184 176
204 10 149 155 132 50 62 156 204 55 126 213 137 67 177 35 231 251 240 50 47 44 164 165 182 197 20 242 80 222 176 171 199 42 137 229 74 149 123 26 80 127 252 55 198 124 227 137
237 215 171 151 150 163 92 170 123 46 4 249 56 24 102 103 153 219 153 132 33 26 69 114 34 213 53 195 100 40 119 16 41 154 122 219 60 145 222 73 214 43 188 225 148 45 205 108 2
71 213 187 26 180 185 53 14 11 228 33 207 76 37 236 3 233 42 76 97 240 208 7 181 179 60 116 163 20 201 31 21 196 254 5 117 18 27 189 35 210 245 208 239 14 208 17 142 95 154 20
```

Алгоритм 8. RSA

Описание алгоритма:

Основные параметры:

На вход:

c – количество необходимых сгенерированных чисел

n – модуль

e – степень

x_0 - начальное значение

w – количество генерируемых бит за шаг

l – длина выходной двоичной последовательности

1. Выбрать случайное целое x_0 .

2. count = (c * l) / w

For i = 1 to count do

a. $x_i \leftarrow x_{x-1}^e \bmod n$

b. В поток добавляем w бит из сгенерированного числа

3. “Нарезаем” последовательность по l бит $z_1, z_2, ..., z_l$

4. Переводим последовательность $z_1, z_2, ..., z_l$ в десятичное число.

Параметры запуска программы:

/g:rsa /n:10000 /f:out.txt /i:100151;224951;287;22528;22

Параметры алгоритма i:

Модуль n , число e , w , начальное значение x .

e удовлетворяет условиям: $1 < e < (p-1)(q-1)$, $\text{НОД}(e, (p-1)(q-1)) = 1$, где $p \cdot q = n$.

x из интервала $[1, n]$

w – длина слова.

Исходный текст алгоритма:

```
def rsa(n_, parameters_):
    p, q, e, x, l = int(parameters_[0]),
int(parameters_[1]), int(parameters_[2]), int(parameters_[3]), \
int(parameters_[4])

    res = []
    bar = IncrementalBar('Выполнение:', max=n_)
    n = p * q
    f = (p - 1) * (q - 1)
    for i in range(n_):
        counter = 1 - 1
        seqElem = 0
        for j in range(l):
            x = x ** e % n
            bit = x & 1
            seqElem = seqElem | (bit << counter)
            counter -= 1
        res.append(seqElem)
        bar.next()
    bar.finish()
    return res
```


2. $z_i \leftarrow$ последний значащий бит x_i
3. Вернуть z_1, z_2, \dots, z_l .
4. Перевести последовательность бит z_1, z_2, \dots, z_l в десятичное число.

Параметры запуска программы:

/g:bbs /n:10000 /i:7 /f:out.txt

Начальное значение x (взаимно простое с n).

Исходный текст алгоритма:

```
def bbs(n_, parameters_):
    bar = IncrementalBar('Выполнение:', max=n_)
    res = []
    if parameters_[0]:
        x = int(parameters_[0])
    else:
        x = 7
    n, w = 50621, 10      #16637
    for i in range(n_):
        x_bin = ''
        for j in range(w):
            x = (x * x) % n
            x_bin += str(x % 2)
        res.append(int(x_bin, 2))
        bar.next()
    bar.finish()
    return res
```

Результат работы программы:

