

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
ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

студента 4 курса 431 группы

факультета компьютерных наук и информационных технологий

Никитина Арсения Владимировича

Научный руководитель

Доцент

подпись, дата

И. И. Слеповичев

Саратов 2023

Задание 2. Преобразование ПСЧ к заданному распределению.

На вход подается псевдослучайная последовательность, сгенерированная RSA.

Для управления программой передаются параметры:

/f:<имя_файла> – имя файла с входной последовательностью;

/d:<распределение> – код распределения для преобразования последовательности.

1. st – стандартное равномерное с заданным интервалом;
2. tr – треугольное распределение;
3. ex – общее экспоненциальное распределение;
4. nr – нормальное распределение;
5. gm – гамма распределение;
6. ln – логнормальное распределение;
7. ls – логистическое распределение;
8. bi – биномиальное распределение.

/m:<число> – модуль;

/r:<параметр1> – 1-й параметр, необходимый для генерации ПСЧ заданного распределения;

/q:<параметр2> – 2-й параметр, необходимый для генерации ПСЧ заданного распределения;

/w:<параметр3> – 3-й параметр, необходимый для генерации ПСЧ заданного распределения (для гамма распределения);

/o:<имя_файла> – имя файла с выходной последовательностью.

Алгоритм 1. Стандартное равномерное с заданным интервалом.

Описание алгоритма.

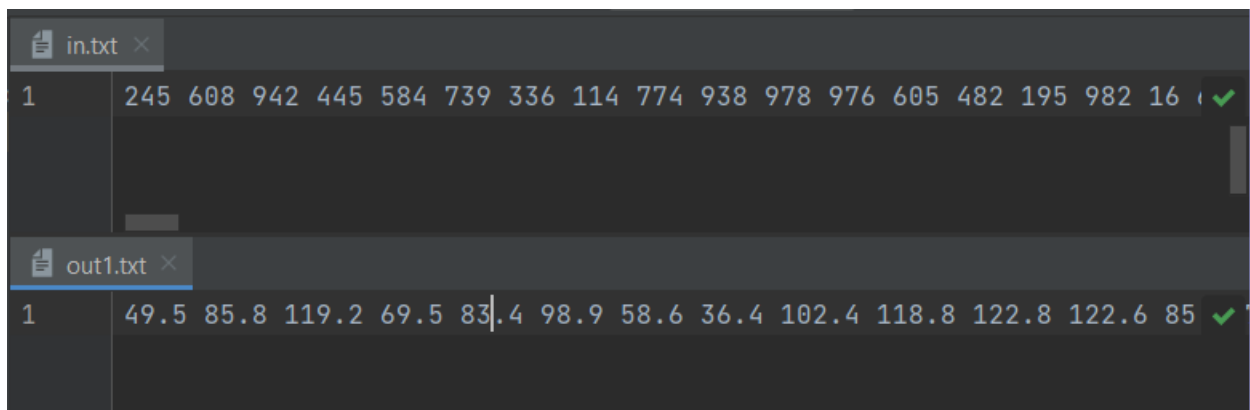
Преобразование:

$st(x, p, q, m) = p + U(x, m) * q$, где $U(x, m) = x / m$.

Ограничения на параметры: $m > 0$, любое входное число последовательности $< m$. Ограничений на p, q нет.

Пример запуска программы.

```
PS D:\rnc> python main.py /d:st /f:in.txt /o:out1.txt /m:1000 /p:25 /q:100
```



Исходный текст программы находится в файле `st.py` и приведен в Приложении А.

Алгоритм 2. Треугольное распределение.

Описание алгоритма.

Преобразование:

$tr(x_1, x_2, p, q, m) = p + q * (U(x_1, m) + U(x_2, m) - 1)$

Ограничения на параметры: $m > 0$, любое входное число последовательности $< m$. Ограничений на p, q нет.

Пример запуска программы.

```
PS D:\rnc> python main.py /d:tr /f:in.txt /o:out2.txt /m:1000 /p:0 /q:1
```

in.txt	
1	245 608 942 445 584 739 336 114 774 938 978 976 605 482 195 982 16 ✓
out2.txt	
1	-0.147 0.387 0.323 -0.515 0.712 0.954 0.087 0.177 -0.299 -0.221 -0.31 ✓

Исходный текст программы находится в файле `tr.py` и приведен в Приложении А.

Алгоритм 3. Общее экспоненциальное распределение.

Описание алгоритма.

Преобразование:

$$ex(x, p, q, m) = p - q * \ln(U(x, m))$$

Ограничения на параметры: $m > 0$, любое входное число последовательности $< m$. Ограничений на p, q нет.

Пример запуска программы.

```
PS D:\rnc> python main.py /d:ex /f:in.txt /o:out3.txt /m:1000 /p:0 /q:1
```

in.txt	
1	245 608 942 445 584 739 336 114 774 938 978 976 605 482 195 982 16 ✓
out3.txt	
1	1.406 0.498 0.06 0.81 0.538 0.302 1.091 2.172 0.256 0.064 0.022 0.01 ✓

Исходный текст программы находится в файле `ex.py` и приведен в Приложении А.

Алгоритм 4. Нормальное распределение.

Описание алгоритма.

Преобразование:

$y_1, y_2 = nr(x_1, x_2, p, q):$

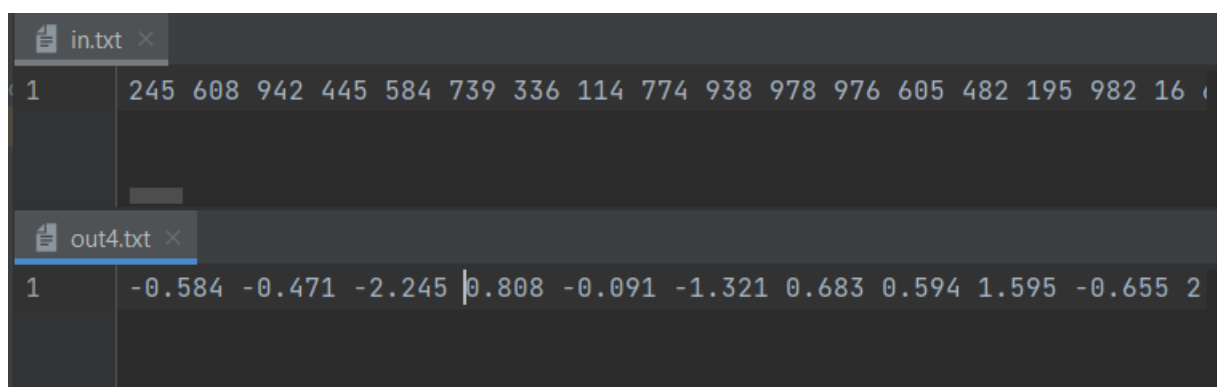
$$y_1 = p + q * \sqrt{(-2 \ln(1 - U(x_1)))} \cos(2 * \pi * U(x_2))$$

$$y_2 = p + q * \sqrt{(-2 \ln(1 - U(x_1)))} \sin(2 * \pi * U(x_2))$$

Ограничения на параметры: $m > 0$, любое входное число последовательности $< m$. Ограничений на p, q нет.

Пример запуска программы.

```
PS D:\rnc> python main.py /d:nr /f:in.txt /o:out4.txt /m:1000 /p:0 /q:1
```



Исходный текст программы находится в файле `nr.py` и приведен в Приложении А.

Алгоритм 5. Гамма распределение.

Описание алгоритма.

Преобразование:

При $w = k$ при k принадлежит $Z, k > 0$

$$y = gamma_k(x_1, \dots, x_w, p, q, w, m) = p - q * \ln([1 - U(x_1, m)] * \dots * [1 - U(x_m)])$$

При $w = k + 0.5$ при k принадлежит $Z, k \geq 0$

$$y_1, y_2 = gamma_{k+0.5}(x_1, \dots, x_k, x_{k+1}, \dots, x_{2k}, x_{2k+1}, x_{2k+2}, p, q, m):$$

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

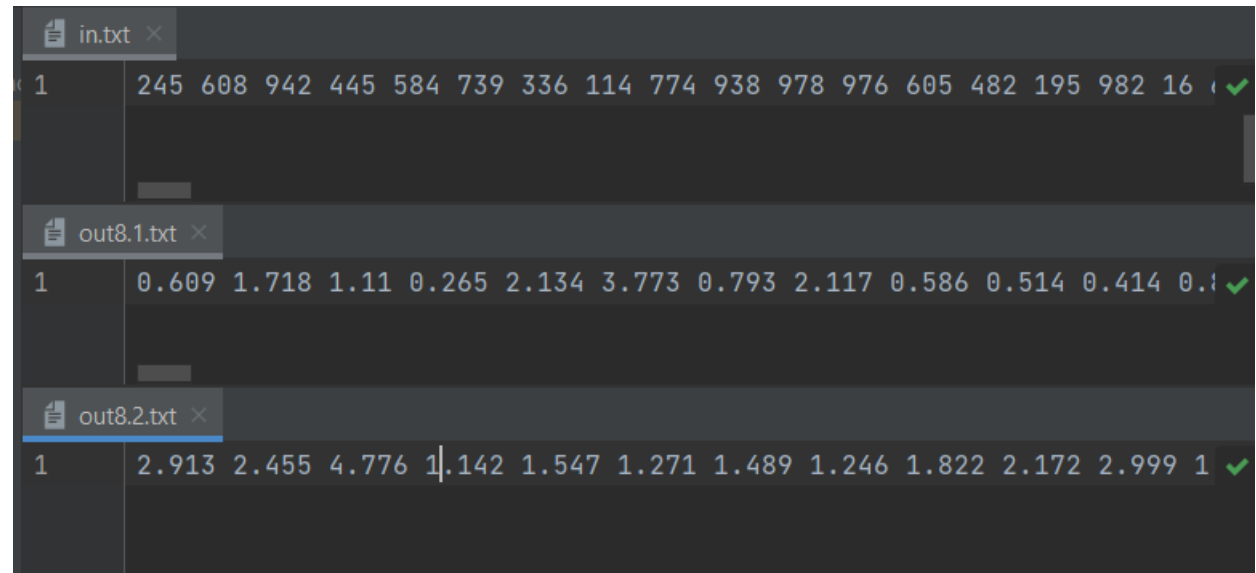
$$y_1 = p + q * (z^2/2 - \ln([1 - U(x_3, m)] * \dots * [1 - U(x_k + 2, m)]))$$

$$y_2 = p + q * (z^2/2 - \ln([1 - U(x_{k+3}, m)] * \dots * [1 - U(x_k + 2, m)]))$$

Ограничения на параметры: $m > 0$, любое входное число последовательности $< m$. Ограничений на p, q, w нет.

Пример запуска программы.

```
PS D:\rnc> python main.py /d:gm /f:in.txt /o:out8.1.txt /m:1000 /p:0 /q:0.5 /w:2
PS D:\rnc> python main.py /d:gm /f:in.txt /o:out8.2.txt /m:1000 /p:0 /q:0.5 /w:4.5
```



File	Line 1	Line 2	Line 3	Line 4	Line 5	Line 6	Line 7	Line 8	Line 9	Line 10	Line 11	Line 12	Line 13	Line 14	Line 15	Line 16
in.txt	1	245	608	942	445	584	739	336	114	774	938	978	976	605	482	195
out8.1.txt	1	0.609	1.718	1.11	0.265	2.134	3.773	0.793	2.117	0.586	0.514	0.414	0.314	0.214	0.114	0.014
out8.2.txt	1	2.913	2.455	4.776	1.142	1.547	1.271	1.489	1.246	1.822	2.172	2.999	1			

Исходный текст программы находится в файле gm.py и приведен в Приложении А.

Алгоритм 6. Логнормальное распределение.

Описание алгоритма.

Преобразование:

$$y_1, y_2 = \text{lognorm}(x_1, x_2, p, q): z_1, z_2 = \text{norm}(x_1, x_2, 0, 1, m) y_1 = p + \exp(q - z_1)$$

$$y_2 = p + \exp(q - z_2)$$

Ограничения на параметры: $m > 0$, любое входное число последовательности $< m$. Ограничений на p, q нет.

Пример запуска программы.

```
PS D:\rnc> python main.py /d:\ln /f:in.txt /o:out5.txt /m:1000 /p:-5 /q:2
```

```
in.txt x
1 245 608 942 445 584 739 336 114 774 938 978 976 605 482 195 982 16 ✓
```

```
out5.txt x
1 8.245 6.829 64.774 -1.707 3.097 22.696 -1.266 -0.921 -3.501 9.225 - ✓
```

Исходный текст программы находится в файле ln.py и приведен в Приложении А.

Алгоритм 7. Логистическое распределение.

Описание алгоритма.

Преобразование:

$y = \text{logistic}(x, p, q, m):$

$u = U(x, m)$

$y = p + q * \ln(u / (1 - u))$

Ограничения на параметры: $m > 0$, любое входное число последовательности $< m$. Ограничений на p, q нет.

Пример запуска программы.

```
PS D:\rnc> python main.py /d:\ls /f:in.txt /o:out6.txt /m:1000 /p:0 /q:1
```

```
in.txt x
1 245 608 942 445 584 739 336 114 774 938 978 976 605 482 195 982 16 ✓
```

```
out6.txt x
1 -1.125 0.439 2.788 -0.221 0.339 1.041 -0.681 -2.051 1.231 2.717 3.7 ✓
```

Исходный текст программы находится в файле ls.py и приведен в Приложении А.

Алгоритм 8. Биномиальное распределение.

Описание алгоритма.

Преобразование:

$y = \text{binominal}(x, p, q, m):$

$u = U(x)$

$s = 0$

$k = 0$

loopstart:

$s = s + C_q^k p^k (1-p)^{q-k}$

if $s > u$:

$y = k$

Завершить

if $k < q-1$:

$k = k+1$

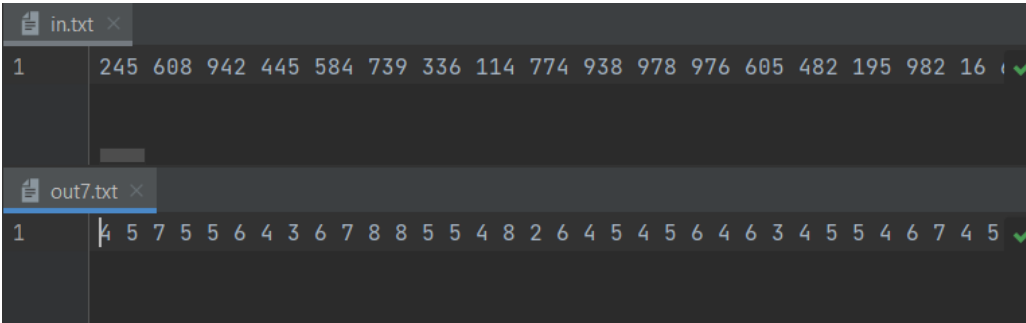
перейти к loopstart

$y = q$

Ограничения на параметры: $m > 0$, любое входное число последовательности $< m$. Ограничений на p, q нет.

Пример запуска программы.

```
PS D:\rnc> python main.py /d:bi /f:in.txt /o:out7.txt /m:1000 /p:0.5 /q:10
```



The screenshot shows a file explorer with two files: `in.txt` and `out7.txt`. The `in.txt` file contains a sequence of 1000 numbers generated by the binomial distribution algorithm. The `out7.txt` file contains a sequence of 1000 numbers generated by the binomial distribution algorithm.

Исходный текст программы находится в файле bi.py и приведен в Приложении А.

ПРИЛОЖЕНИЕ А

Листинг программы

1) Файл st.py

```
import help

def st(x, a, b, m):
    rez = a + (x / m) * b
    return rez

def main(f_name_in, f_name_out, args):
    try:
        f_in = help.op_file(f_name_in)
        m, a, b = help.get_arg(args)
        f = open(f_name_out, 'w')
        for i in range(0, len(f_in)):
            f.write(str(round(st(f_in[i], a, b, m), 3)) + " ")
        f.close()
    except FileNotFoundError:
        print('Файл не найден\n')
        exit(0)
    except:
        print('Ошибка\n')
        exit(0)
```

2) Файл tr.py

```
import help

def tr(x1, x2, a, b, m):
    rez = a + b * ((x1 / m) + (x2 / m) - 1)
    return rez

def main(f_name_in, f_name_out, args):
    try:
        f_in = help.op_file(f_name_in)
        m, a, b = help.get_arg(args)
        f = open(f_name_out, 'w')
        i = 0
        while i < len(f_in) - 1:
            f.write(str(round(tr(f_in[i], f_in[i + 1], a, b, m), 3)) + " ")
            i = i + 2
        f.close()
    except FileNotFoundError:
        print('Файл не найден\n')
        exit(0)
    except:
        print('Ошибка\n')
        exit(0)
```

3) Файл ex.py

```
import math, help

def ex(x, a, b, m):
```

```

    rez = a - b * math.log(x / m)
    return rez

def main (f_name_in, f_name_out, args):
    try:
        f_in = help.op_file(f_name_in)
        m, a, b = help.get_arg(args)
        f = open(f_name_out, 'w')
        for i in range(0, len(f_in)):
            f.write(str(round(ex(f_in[i], a, b, m), 3)) + " ")
        f.close()
    except FileNotFoundError:
        print('Файл не найден\n')
        exit(0)
    except :
        print('Ошибка\n')
        exit(0)

```

4) Файл nr.py

```

import math, help

def nr(x1, x2, a, b, m):
    rez = []
    y1 = a + b * math.sqrt(math.fabs(-2 * math.log(1 - (x1 / m)))) *
math.cos(2*math.pi*(x2 / m))
    y2 = a + b * math.sqrt(math.fabs(-2 * math.log(1 - (x1 / m)))) *
math.sin(2*math.pi*(x2 / m))
    rez.append(y1)
    rez.append(y2)
    return rez

def main (f_name_in, f_name_out, args):
    try:
        f_in = help.op_file(f_name_in)
        m, a, b = help.get_arg(args)
        f = open(f_name_out, 'w')
        i = 0
        while i < len(f_in) - 1:
            rez = nr(f_in[i], f_in[i + 1], a, b, m)
            f.write(str(round(rez[0], 3)) + " " + str(round(rez[1], 3)) + "
")
            i = i + 2
        f.close()
    except FileNotFoundError:
        print('Файл не найден\n')
        exit(0)
    except:
        print('Ошибка\n')
        exit(0)

```

5) Файл gm.py

```

import math, help, nr

def gamma(x, a, b, c, fl_c, m):
    if fl_c:
        umn = 1

```

```

        for key in x:
            umn = umn * (1 - (key / m))
        y = a - b * math.log(math.fabs(umn))
        return str(round(y, 3))
    else:
        k = int(c - 0.5)
        umn = 1
        z1, z2 = nr.nr(x[0], x[1], 0, 1, m)
        for key in range(2, k + 2):
            umn = umn * (1 - (x[key] / m))
        y1 = a + b * ( ((z1 ** 2) / 2) - math.log(umn))
        umn = 1
        for key in range(k + 2, len(x)):
            umn = umn * (1 - (x[key] / m))
        y2 = a + b * (((z2 ** 2) / 2) - math.log(umn))
        return str(round(y1, 3)) + " " + str(round(y2, 3))

def func_chunks_generators(lst, n):
    for i in range(0, len(lst), n):
        yield lst[i : i + n]

def main (f_name_in, f_name_out, args):
    try:
        f_in = help.op_file(f_name_in)
        m, a, b, c, fl_c = help.get_arg_gm(args)
        f = open(f_name_out, 'w')
        if fl_c:
            mass = list(func_chunks_generators(f_in, c))
            for i in range(0, len(mass)):
                f.write(gamma(mass[i], a, b, c, fl_c, m) + " ")
        else:
            k = int(c - 0.5)
            mass = list(func_chunks_generators(f_in, 2 * k + 2))
            for i in range(0, len(mass)):
                f.write(gamma(mass[i], a, b, c, fl_c, m) + " ")
        f.close()

    except FileNotFoundError:
        print('Файл не найден\n')
        exit(0)
    except:
        print('Ошибка\n')
        exit(0)

```

6) Файл ln.py

```
import math, help, nr
```

```

def obr (s):
    s = s[2:5]
    return s

def lnr(x1, x2, a, b, m):
    rez = []
    tmp = nr.nr(x1, x2, 0, 1, m)
    y1 = a + math.exp(b - tmp[0])
    y2 = a + math.exp(b - tmp[1])
    rez.append (y1)
    rez.append (y2)
    return rez

```

```

def main (f_name_in, f_name_out, args):
    try:
        f_in = help.op_file(f_name_in)
        m, a, b = help.get_arg(args)
        f = open(f_name_out, 'w')
        i = 0
        while i < len(f_in) - 1:
            rez = lnr(f_in[i], f_in[i + 1], a, b, m)
            f.write(str(round(rez[0], 3)) + " " + str(round(rez[1], 3)) + "
")
            i = i + 2
        f.close()
    except FileNotFoundError:
        print('Файл не найден\n')
        exit(0)
    except:
        print('Ошибка\n')
        exit(0)

```

7) Файл ls.py

```
import math, help
```

```

def logistic(x, a, b, m):
    u = x / m
    rez = a + b * math.log(u/(1-u))
    return rez

def main (f_name_in, f_name_out, args):
    try:
        f_in = help.op_file(f_name_in)
        m, a, b = help.get_arg(args)
        f = open(f_name_out, 'w')
        for i in range(0, len(f_in)):
            f.write(str(round(logistic(f_in[i], a, b, m), 3)) + " ")
        f.close()
    except FileNotFoundError:
        print('Файл не найден\n')
        exit(0)
    except :
        print('Ошибка\n')
        exit(0)

```

8) Файл bi.py

```
import math, help
```

```

def binominal(x,a,b,m):
    u = x / m
    s = 0
    y = 0
    for k in range(b):
        c_b = math.factorial(b)/(math.factorial(k) * math.factorial(b - k))
        s += c_b * (a ** k) * ((1 - a) ** (b - k))
        if s > u:
            y = k
            break
    y = b
    k += 1

```

```

        return y

def main (f_name_in, f_name_out, args):
    try:
        f_in = help.op_file(f_name_in)
        m, a, b = help.get_arg(args)
        f = open(f_name_out, 'w')
        for i in range(0, len(f_in)):
            f.write(str(binominal(f_in[i], a, b, m)) + " ")
        f.close()
    except FileNotFoundError:
        print('Файл не найден\n')
        exit(0)
    except :
        print('Ошибка\n')
        exit(0)

```

9) Управляющий файл main.py

```

import re
from sys import stdout, argv
import st, tr, ex, nr, ln, ls, bi, gm
mass_code = ["st", "tr", "ex", "nr", "ln", "ls", "bi", "gm"]

def helps():
    print("""
1. st - стандартное равномерное с заданным интервалом;
2. tr - треугольное распределение;
3. ex - общее экспоненциальное распределение;
4. nr - нормальное распределение;
5. gm - гамма распределение;
6. ln - логнормальное распределение;
7. ls - логистическое распределение;
8. bi - биномиальное распределение.

/m:<число> - модуль;
/p:<параметр1> - 1-й параметр, необходимый для генерации ПСЧ заданного
распределения;
/q:<параметр2> - 2-й параметр, необходимый для генерации ПСЧ заданного
распределения;
/w:<параметр3> - 3-й параметр, необходимый для генерации ПСЧ заданного
распределения (для гамма распределения);
""")

def prov_code(mass):
    mass_code = ["st", "tr", "ex", "nr", "gm", "ln", "ls", 'bi']
    code = mass[0]
    code = code[3:]
    try:
        if mass_code.index(code) != -1:
            return code
    except ValueError:
        print("Ошибка, введите нужный код распределения")

def file_name(strok):
    file = strok
    file = file[3:]
    return file

```

```

def file_resurs(mass):
    file_in = ""
    file_out = ""
    rez = []
    if mass[0].find("/f:") != -1:
        file_in = file_name(mass[0])
    else:
        if mass[0].find("/o:") != -1:
            file_out = file_name(mass[0])
    if mass[1].find("/o:") != -1:
        file_out = file_name(mass[1])
    rez.append(file_in)
    rez.append(file_out)
    if (file_in != "" ) and (file_out != "" ):
        rez.append(2)
    else:
        if ((file_in == "" ) and (file_out != "" )) or ((file_in != "" ) and
(file_out == "")):
            rez.append(1)
        else:
            rez.append(0)
    return rez

```

```

b = argv
a = []
a.append(b[0])
cnt = 0
for i in range(0, len(b), 1):
    if b[i].startswith('/h') == 1:
        a.append(b[i])
for i in range(0, len(b), 1):
    if b[i].startswith('/d:') == 1:
        a.append(b[i])
        cnt += 1
for i in range(0, len(b), 1):
    if b[i].startswith('/f:') == 1:
        a.append(b[i])
        cnt += 1
for i in range(0, len(b), 1):
    if b[i].startswith('/o:') == 1:
        a.append(b[i])
        cnt += 1
for i in range(0, len(b), 1):
    if b[i].startswith('/m:') == 1:
        a.append(b[i])
        cnt += 1
for i in range(0, len(b), 1):
    if b[i].startswith('/p:') == 1:
        a.append(b[i])
        cnt += 1
for i in range(0, len(b), 1):
    if b[i].startswith('/q:') == 1:
        a.append(b[i])
        cnt += 1
for i in range(0, len(b), 1):
    if b[i].startswith('/w:') == 1:
        a.append(b[i])
        cnt += 1

```

```

try:
    args = []
    consol_in = False

```

```

consol_out = False
file_name_in = ""
del a[0]
if a[0] == '/h':
    helps()
    exit(0)
code = prov_code(a)
if code != "":
    del a[0]
if len(a) >= 2:
    f_res = file_resurs(a)
else:
    f_res = ["", "", 0]
file_in, file_out, count_udal = f_res
for i in range (int(count_udal)):
    del a[0]
for i in a:
    args_t = "".join(re.findall(r'[-]?\\d+[.]?[\\d+]*', i))
    args.append(args_t)
if file_in == "":
    print("Не указан входной файл")
    exit(0)
if file_out == "":
    file_out = "rnd.dat"
if code == "st":
    if len(args) != 3:
        stdout.write('Ошибка2\\n')
        exit(0)
    st.main(file_in, file_out, args)
if code == "tr":
    if len(args) != 3:
        stdout.write('Ошибка3\\n')
        exit(0)
    tr.main(file_in, file_out, args)
if code == "ex":
    if len(args) != 3:
        stdout.write('Ошибка4\\n')
        exit(0)
    ex.main(file_in, file_out, args)
if code == "nr":
    if len(args) != 3:
        stdout.write('Ошибка5\\n')
        exit(0)
    nr.main(file_in, file_out, args)
if code == "ln":
    if len(args) != 3:
        stdout.write('Ошибка6\\n')
        exit(0)
    ln.main(file_in, file_out, args)
if code == "ls":
    if len(args) != 3:
        stdout.write('Ошибка7\\n')
        exit(0)
    ls.main(file_in, file_out, args)
if code == "bi":
    if len(args) != 3:
        stdout.write('Ошибка8\\n')
        exit(0)
    bi.main(file_in, file_out, args)
if code == "gm":
    if len(args) != 4:
        stdout.write('Ошибка9\\n')
        exit(0)
    gm.main(file_in, file_out, args)

```

```
except FileNotFoundError:
    stdout.write('Файл не найден\n')
    exit(0)
```

10) Вспомогательный файл

```
def op_file(file_name):
    fale = open(file_name, "r")
    rez = fale.read().split()
    for x in rez:
        if x.find(".") != -1:
            print("Не верные входные значения")
            exit(0)
    fale.close()
    return[int(elem) for elem in rez]
```

```
def get_arg(args):
    m = int(args[0])
    if args[1].find(".") != -1:
        a = float(args[1])
    else:
        a = int(args[1])
    if args[2].find(".") != -1:
        b = float(args[2])
    else:
        b = int(args[2])
    return m, a, b
```

```
def get_arg_gm(args):
    fl_c = False
    m = int(args[0])
    if args[1].find(".") != -1:
        a = float(args[1])
    else:
        a = int(args[1])
    if args[2].find(".") != -1:
        b = float(args[2])
    else:
        b = int(args[2])
    if args[3].find(".") != -1:
        if args[3][-1] != '5':
            print("Ошибка в параметре w")
            exit(0)
        c = float(args[3])
        if c < 0:
            print("Ошибка в параметре w")
            exit(0)
    else:
        c = int(args[3])
        if c <= 0:
            print("Ошибка в параметре w")
            exit(0)
    fl_c = True
    return m, a, b, c, fl_c
```