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

**Преобразование ПСЧ к заданному распределению**

ОТЧЕТ ПО ДИСЦИПЛИНЕ

«ТЕОРИЯ ПСЕВДОСЛУЧАЙНЫХ ГЕНЕРАТОРОВ»

студента 4 курса 431 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Ухова Александра Андреевича

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Научный руководитель  Ст. преподаватель | \_\_\_\_\_\_\_\_\_\_ | И.И. Слеповичев |
|  | подпись, дата |  |

Саратов 2024

**СОДЕРЖАНИЕ**

[1 Описание технической задачи 3](#_gjdgxs)

[2 Стандартное равномерное с заданным интервалом 5](#_30j0zll)

[3 Треугольное распределение 8](#_3znysh7)

[4 Общее экспоненциальное распределение 10](#_2et92p0)

[5 Нормальное распределение 12](#_tyjcwt)

[6 Гамма распределение (для параметра *c*=*k*) 15](#_3dy6vkm)

[7 Логнормальное распределение 18](#_1t3h5sf)

[8 Логистическое распределение 20](#_4d34og8)

[9 Биномиальное распределение 22](#_2s8eyo1)

[ПРИЛОЖЕНИЕ А 25](#_17dp8vu)

# **Описание технической задачи**

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.

Название программы: **rnc.exe**

**На входе**

Текстовый файл с десятичными числами (разделитель – любой), интервал преобразуемых значений, параметры распределения.

Для управления приложением предлагается следующий формат параметров командной строки:

/f:<имя\_файла> - имя файла с входной последовательностью.

/d:<распределение> - код распределения для преобразования последовательности. Рекомендуется использовать следующие коды распределений:

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.

/p1:<параметр1> - 1-й параметр, необходимый, для генерации ПСЧ заданного распределения.

/p2:<параметр2> - 2-й параметр, необходимый, для генерации ПСЧ заданного распределения.

/p3:<параметр3> - 3-й параметр, необходимый, для генерации ПСЧ гамма-распределением.

**На выходе**

Текстовый файл distr-xx.dat с преобразованными числами, где <xx> – код распределения.

Тестовая последовательность будет находиться в файле generated.txt, которая и будет поступать на вход программам.

# **Стандартное равномерное с заданным интервалом**

Функция плотности вероятности выглядит следующим образом:

Суть метода генерации случайной величины заключается в том, что, если максимальное значение равномерного целого случайного числа равно , для генерации стандартных равномерных случайных чисел необходимо применять следующую формулу:

Поскольку *X* принимает дискретные значения, величина *U* также является дискретной.

Величина *U* принимает 0,0 только в том случае, если . Значение 1 величина не принимает.

В общем случае функция плотности вероятности выглядит так:

где

Если стандартное равномерное случайное число получено методом, установленным в предыдущем параграфе, то равномерное случайное число должно быть получено в соответствии со следующей формулой

**Исходный текст программы**

# 

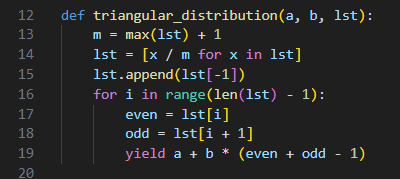
# **Треугольное распределение**

Функция плотности вероятности выглядит следующим образом:

где

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

**Исходный текст программы**

****

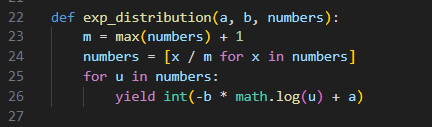
# **Общее экспоненциальное распределение**

Функция плотности вероятности выглядит следующим образом:

где - параметры положения и масштаба экспоненциального распределения соответственно.

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

**Исходный текст программы**

****

# **Нормальное распределение (распределение Гаусса)**

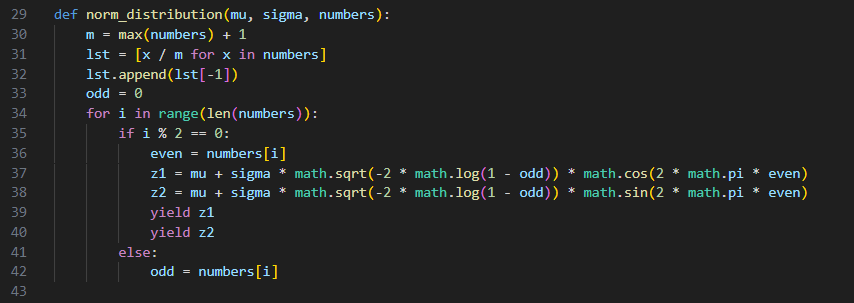
Функция плотности вероятности выглядит следующим образом:

где – среднее и стандартное отклонение нормального распределения соответственно.

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

При получении *U*1, *U*2 линейным конгруэнтным методом последовательно, *U*1 и *U*2 являются зависимыми, таким образом хвост распределений, полученных и может существенно отличаться от истинно нормального распределения.

**Исходный текст программы**

****

# **Гамма распределение (для параметра *c*=*k*)**

Функция гамма-распределения выглядит следующим образом:

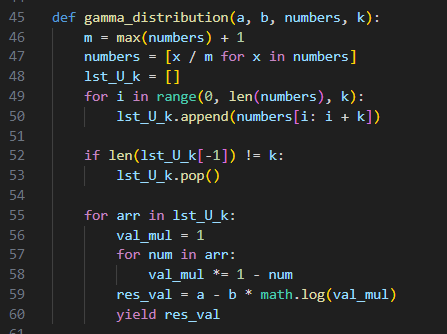
где – параметры положения, масштаба и формы соответственно.

Используя независимые равномерные случайные числа , применяют формулу

.

Этим методом для и может быть получено распределение Хи-квадрат с четным числом степеней свободы.

**Исходный текст программы**

****

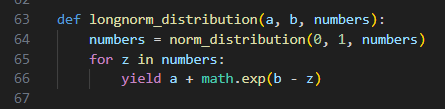
# **Логнормальное распределение**

Функция плотности вероятности выглядит следующим образом:

где – параметры положения и масштаба, соответствующего нормального распределения.

Используя стандартные нормальные случайные числа *Z*, применяют формулу для получения случайных чисел, соответствующих логнормальному распределению.

**Исходный текст программы**

**

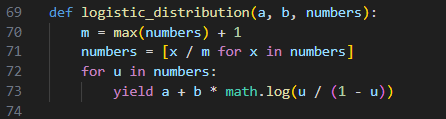
# **Логистическое распределение**

Функция плотности вероятности выглядит следующим образом:

где – параметры положения и масштаба соответственно.

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

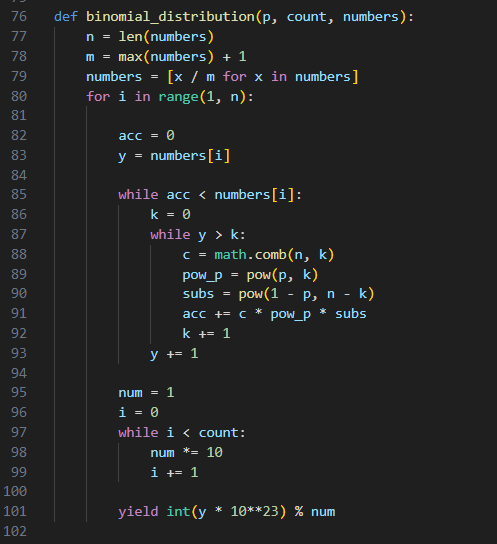
**Исходный текст программы**

****

# **Биномиальное распределение**

Вычисляют функцию распределения

Для получения случайного числа *Y* генерируют стандартное равномерное случайное число *U*. Случайное число *Y* является наименьшим значением *y*, для которого



**ПРИЛОЖЕНИЕ А  
Листинг программы**

import math

import click

def uniform\_distribution(a, b, numbers):

    m = max(numbers) + 1

    for x in numbers:

        yield x / m \* b + a

def triangular\_distribution(a, b, lst):

    m = max(lst) + 1

    lst = [x / m for x in lst]

    lst.append(lst[-1])

    for i in range(len(lst) - 1):

        even = lst[i]

        odd = lst[i + 1]

        yield a + b \* (even + odd - 1)

def exp\_distribution(a, b, numbers):

    m = max(numbers) + 1

    numbers = [x / m for x in numbers]

    for u in numbers:

        yield int(-b \* math.log(u) + a)

def norm\_distribution(mu, sigma, numbers):

    m = max(numbers) + 1

    lst = [x / m for x in numbers]

    lst.append(lst[-1])

    odd = 0

    for i in range(len(numbers)):

        if i % 2 == 0:

            even = numbers[i]

            z1 = mu + sigma \* math.sqrt(-2 \* math.log(1 - odd)) \* math.cos(2 \* math.pi \* even)

            z2 = mu + sigma \* math.sqrt(-2 \* math.log(1 - odd)) \* math.sin(2 \* math.pi \* even)

            yield z1

            yield z2

        else:

            odd = numbers[i]

def gamma\_distribution(a, b, numbers, k):

    m = max(numbers) + 1

    numbers = [x / m for x in numbers]

    lst\_U\_k = []

    for i in range(0, len(numbers), k):

        lst\_U\_k.append(numbers[i: i + k])

    if len(lst\_U\_k[-1]) != k:

        lst\_U\_k.pop()

    for arr in lst\_U\_k:

        val\_mul = 1

        for num in arr:

            val\_mul \*= 1 - num

        res\_val = a - b \* math.log(val\_mul)

        yield res\_val

def longnorm\_distribution(a, b, numbers):

    numbers = norm\_distribution(0, 1, numbers)

    for z in numbers:

        yield a + math.exp(b - z)

def logistic\_distribution(a, b, numbers):

    m = max(numbers) + 1

    numbers = [x / m for x in numbers]

    for u in numbers:

        yield a + b \* math.log(u / (1 - u))

def binomial\_distribution(p, count, numbers):

    n = len(numbers)

    m = max(numbers) + 1

    numbers = [x / m for x in numbers]

    for i in range(1, n):

        acc = 0

        y = numbers[i]

        while acc < numbers[i]:

            k = 0

            while y > k:

                c = math.comb(n, k)

                pow\_p = pow(p, k)

                subs = pow(1 - p, n - k)

                acc += c \* pow\_p \* subs

                k += 1

            y += 1

        num = 1

        i = 0

        while i < count:

            num \*= 10

            i += 1

        yield int(y \* 10\*\*23) % num

def write\_to\_file(numbers, filepath):

    f = open(filepath, "w", encoding="UTF-8")

    data = " ".join(map(str, numbers))

    f.write(data)

    f.close()

def read\_from\_file(filepath):

    f = open(filepath, "r", encoding="UTF-8")

    numbers = map(int, f.read().split())

    return numbers

distributions = {

    'st': uniform\_distribution,

    'tr': triangular\_distribution,

    'ex': exp\_distribution,

    'nr': norm\_distribution,

    'gm': gamma\_distribution,

    'ln': longnorm\_distribution,

    'ls': logistic\_distribution,

    'bi': binomial\_distribution

}

@click.command()

@click.option(

    "/d", help="Указывает тип распределения: "

               "st – стандартное равномерное с заданным интервалом, "

               "tr – треугольное распределение, "

               "ex – общее экспоненциальное распределение, "

               "nr – нормальное распределение, "

               "gm – гамма распределение, "

               "ln – логнормальное распределение, "

               "ls – логистическое распределение, "

               "bi – биномиальное распределение",

    required=True, type=click.Choice(["st", "tr", "ex", "nr", "gm", "ln", "ls", "bi"]))

@click.option("/f", nargs=1, default="generated.dat",

              help="Указывает путь до файла из которого берется входная последовательность чисел")

@click.option("/p1", nargs=1, type=int, required=True,

              help="1-й параметр, необходимый, для генерации ПСЧ заданного распределения")

@click.option("/p2", nargs=1, type=int, required=True,

              help="2-й параметр, необходимый, для генерации ПСЧ заданного распределения (в случае bi указывает на количество разрядов в генерируемом числе до 10^6)")

@click.option("/p3", nargs=1, type=int, help="3-й параметр, необходимый, для генерации ПСЧ гамма-распределением",

              default=None)

def main(d, f, p1, p2, p3):

    try:

        distribution = distributions[d]

        input\_numbers = list(read\_from\_file(f))

        numbers = []

        if p3 == None:

            numbers = [\_ for \_ in distribution(p1, p2, input\_numbers)]

        else:

            numbers = [\_ for \_ in distribution(p1, p2, input\_numbers, p3)]

        write\_to\_file(numbers, f"distr-{d}.dat")

    except Exception as err:

        print(f"В процессе произошла ошибка: {err}")

"""

python rnc.py /d st /p1 134 /p2 123124

python rnc.py /d tr /p1 289 /p2 1323

python rnc.py /d ex /p1 289 /p2 1323

python rnc.py /d nr /p1 234 /p2 12300

python rnc.py /d gm /p1 289 /p2 1323 /p3 2

python rnc.py /d ln /p1 289 /p2 132

python rnc.py /d ls /p1 289 /p2 132

python rnc.py /d bi /p1 234 /p2 5

"""

if \_\_name\_\_ == "\_\_main\_\_":

    main()