**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное бюджетное образовательное учреждение высшего образования**

**Московский авиационный институт**

**(национальный исследовательский университет)**

Институт №3

«Системы управления, информатики и электроэнергетики»

Кафедра 304

**Пояснительная записка к курсовой работе**

по учебной дисциплине «Программирование»

Вариант № 3

Выполнил:

студент группы М30-209Б-21

Викулов Д. Г.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Принял:

доц. каф. 304, к.т.н. Новиков П. В.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2022 г.

Москва 2022

**Содержание**

1. [Задание……………………………………………………………………………………3](#Задание)
2. [Теоретическая часть…………………………………………………………………...…4](#Теория)
3. [Схема алгоритма..…………...……………………………………………………….......5](#Схема)
4. [Код программы…………………………………………….……………………………60](#Код)
5. [Результаты работы программы…………..…………………….…..…………………102](#Результаты)

6[. Объяснение вывода гистограмм и графиков на экран………………..………………174](#Объяснение)

7. [Вывод………………….………………………………………………….……..………183](#Вывод)

8. [Список источников…………………………….…………………….……………..…..185](#Литература)

**Задание**

При выполнении курсовой работы следует продемонстрировать знания и умения в использовании основных синтаксических конструкций императивных (процедурных) языков программирования на примере языка С++, а именно: умение создавать переменные встроенных типов данных, умение использовать встроенные арифметические, логические и иные операции, а также библиотечные функции, умение создавать и применять пользовательские функции и переменные структурированных пользовательских типов данных, умение использовать ввод и вывод данных по формату, а также без формата (в автоматическом формате). Результаты работы программ следует сохранять в текстовых и/или двоичных файлах. Необходимо продемонстрировать умение считывать данные из текстовых и/или двоичных файлов в программу. Также необходимо показать умение отображать в таблицах, диаграммах и графиках полученные при работе созданных программ результаты.

Разработать программу на языке С++ с использованием встроенного генератора псевдослучайных чисел, моделирующую гауссовское распределение с помощью косинусного преобразования Бокса-Мюллера. Разработать функции для создания требуемого нормального распределения и для вычисления его статистических характеристик. Построить гистограммы полученного распределения в зависимости от количества псевдослучайных чисел.

**Теоретическая часть**

**Преобразование Бокса —Мюллера** — метод моделирования стандартных нормально распределённых случайных величин. Метод является точным, в отличие, например, от методов, основывающихся на центральной предельной теореме.

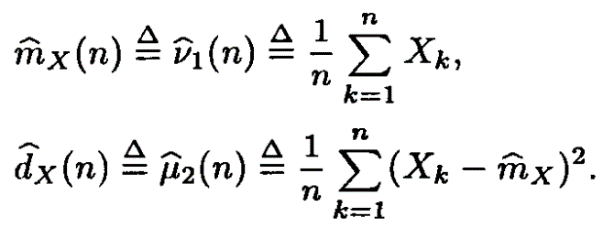
Для равномерно распределённых на отрезке (0;1] случайных величин u и v формула косинусного преобразования Бокса-Мюллера имеет вид:



Путём осуществления преобразования для достаточно большого количества пар равномерно распределённых случайных величин, получим массив выборки.

Введём понятия основных статистических величин полученной выборки:

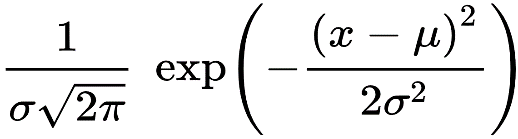
**Выборочным средним** и **выборочной дисперсией** случайной величины X называются соответственно:



**Функции плотностей равномерного** и **нормального** распределений соответственно, использующиеся в программе для расчёта теоретических плотностей распределений:

1. ,

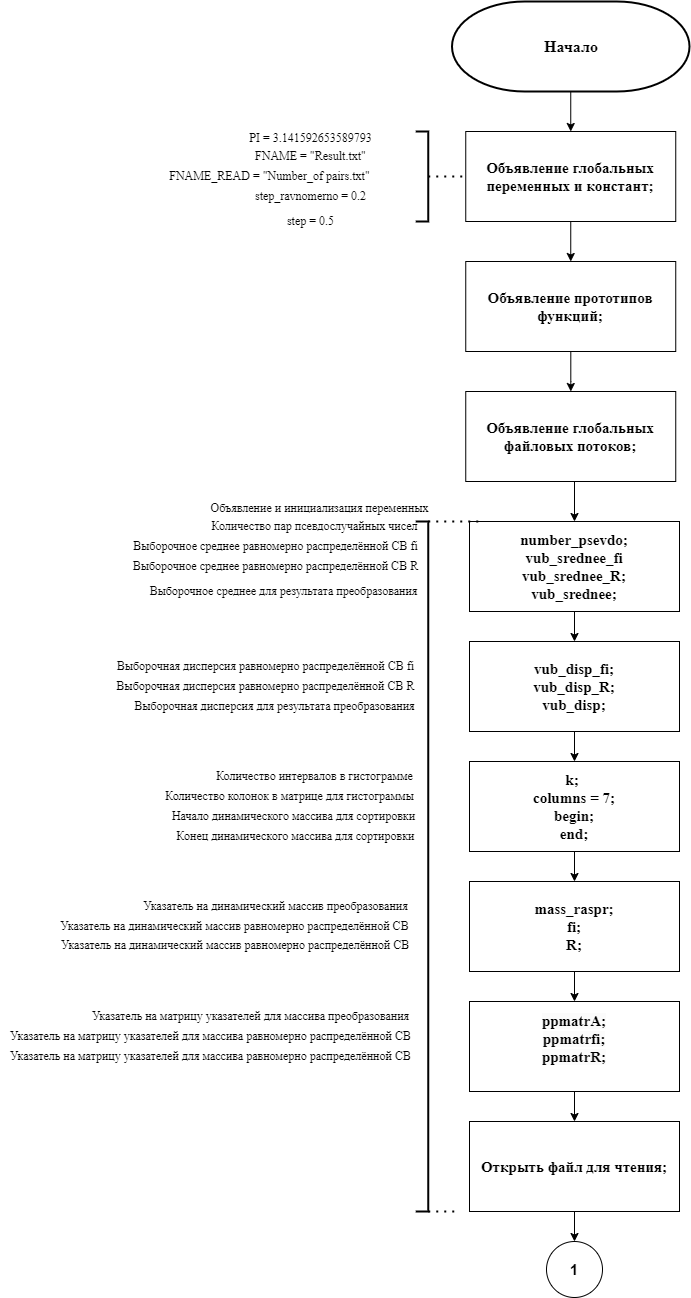
где в нашем случае a=0; b=1

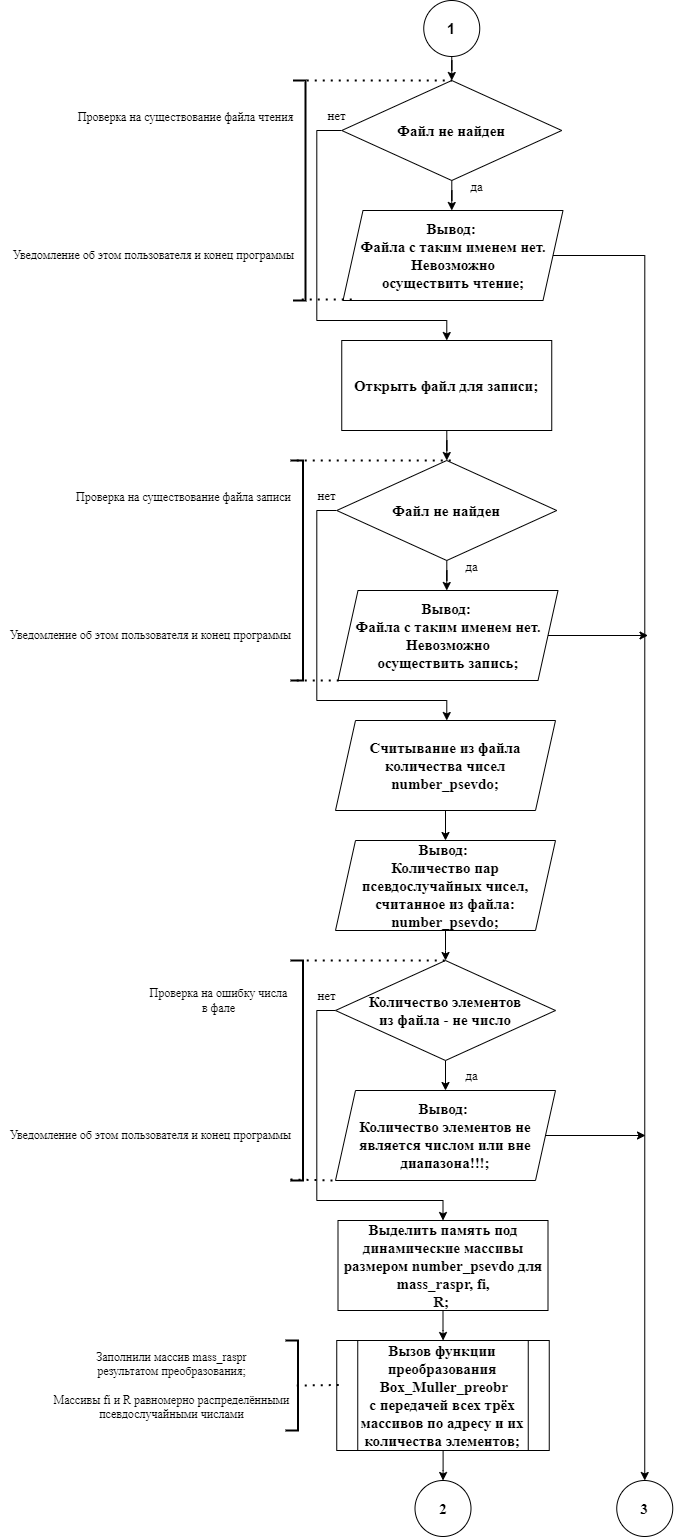
1. ,

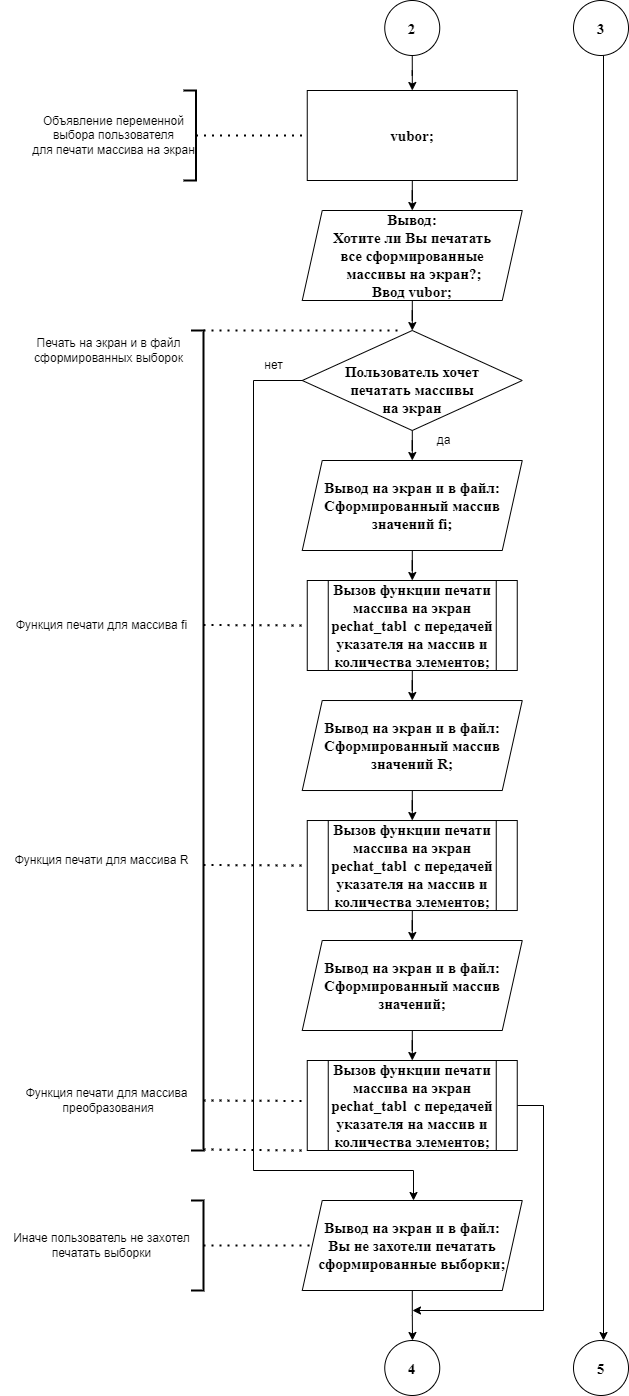
где σ – стандартное отклонение и μ – выборочное среднее

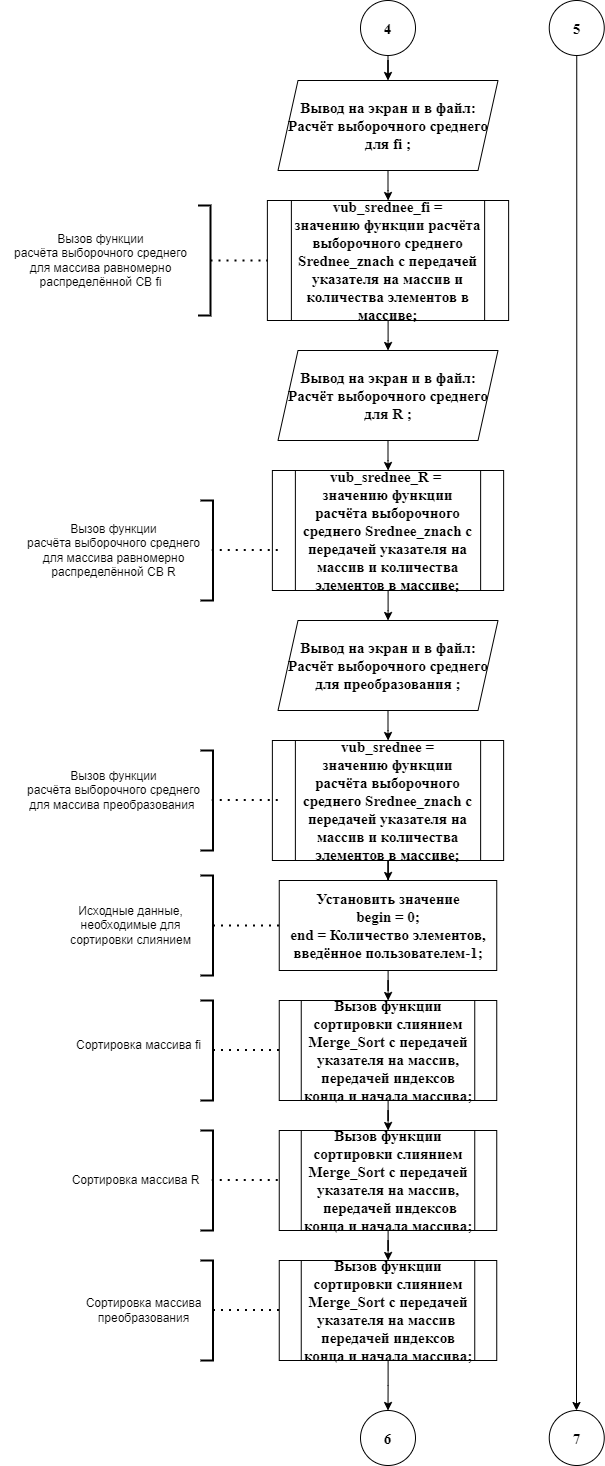
**Схема алгоритма**

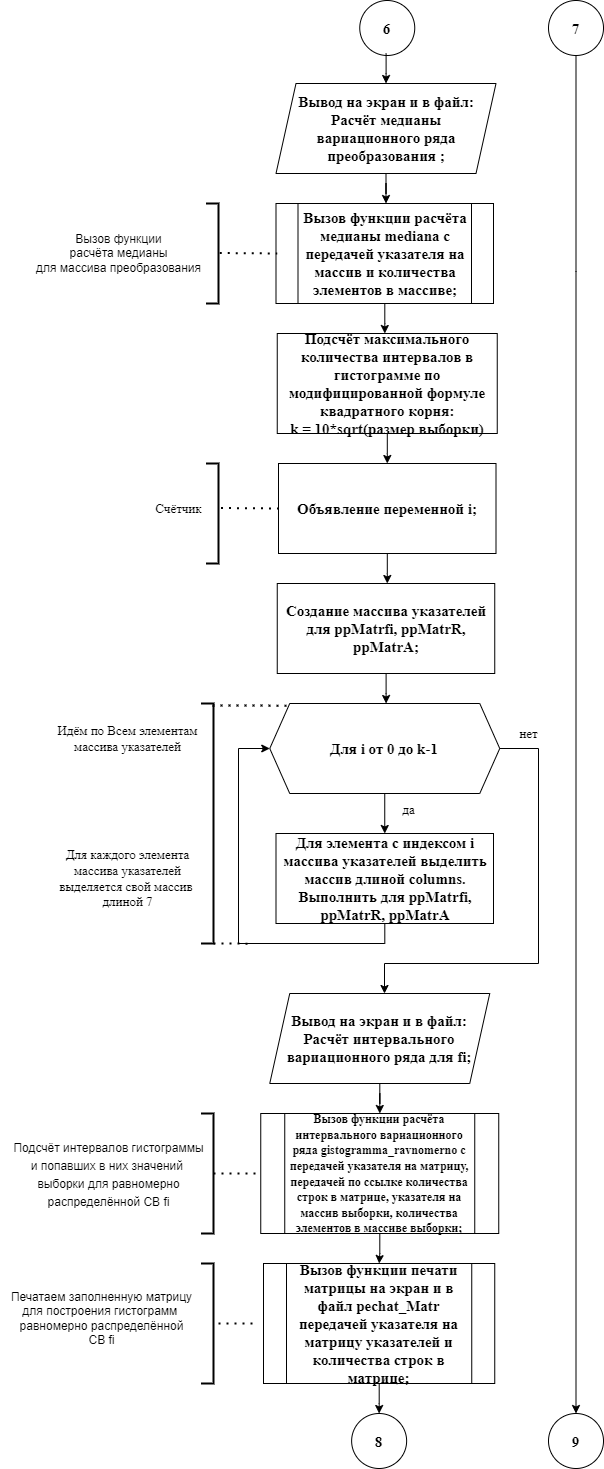
**Схема функции main():**



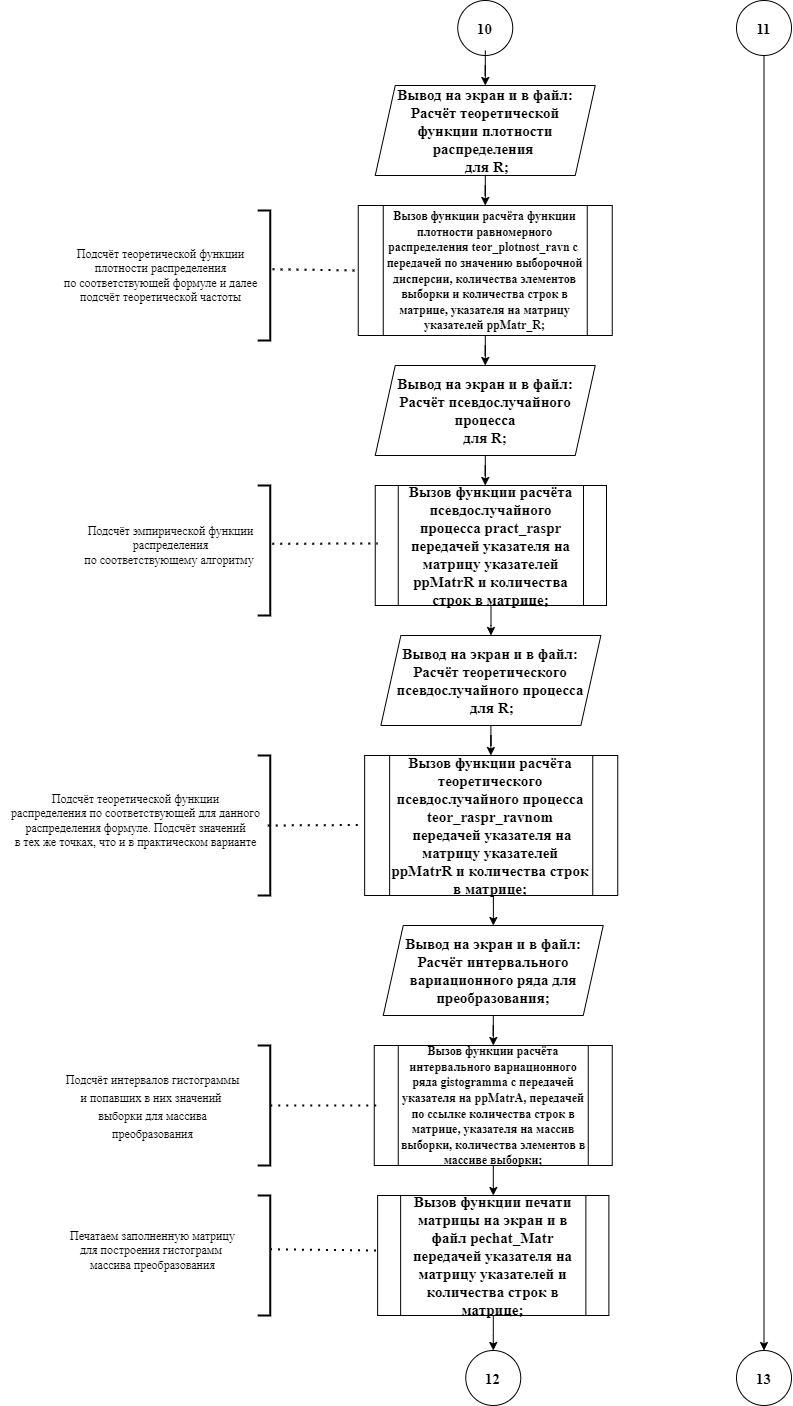


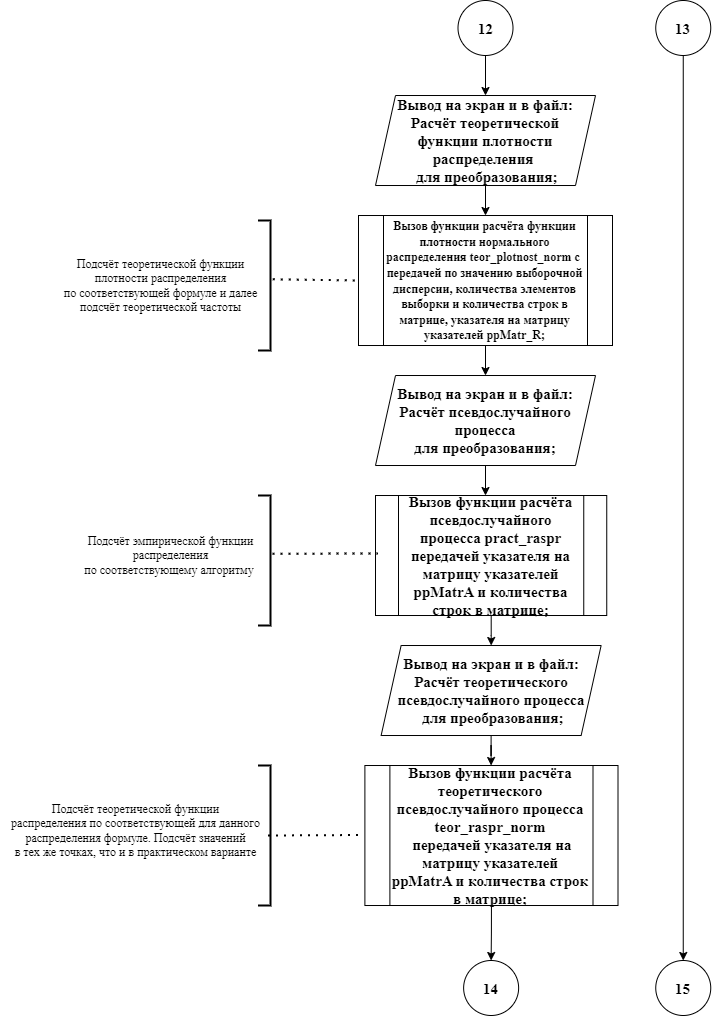


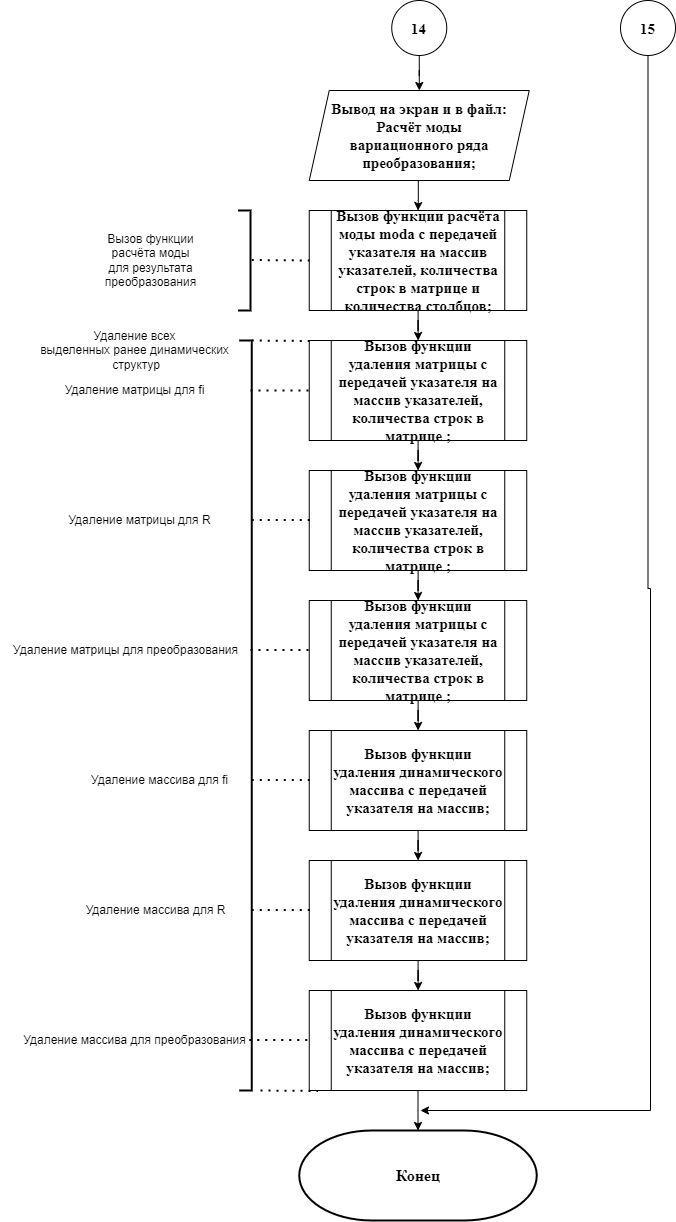












**Описание функции : udalenie**

**1.Назначение**

Удаление динамического массива - вектора

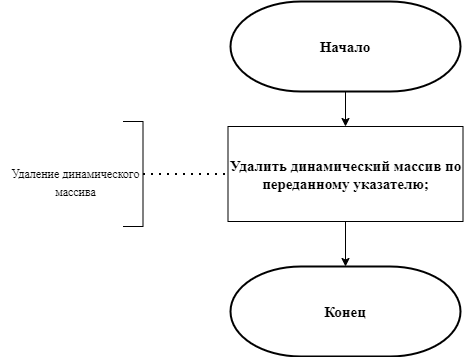
**2.Прототип функции**

void udalenie(double\* mass\_raspr) // указатель на первый элемент / / / / / . // динамического массива-вектора

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной/ Выходной |
| udalenie | void | Удаление динамического массива-вектора | Выходной |
| mass\_raspr | double\* | Указатель на динамический массив-вектор | Входной |

1. **Блок-схема функции udalenie():**



**Описание функции : pechat\_tabl**

**1.Назначение**

Печать сформированного массива в таблице

**2.Прототип функции**

void pechat\_tabl(double\* mass\_raspr // указатель на первый элемент

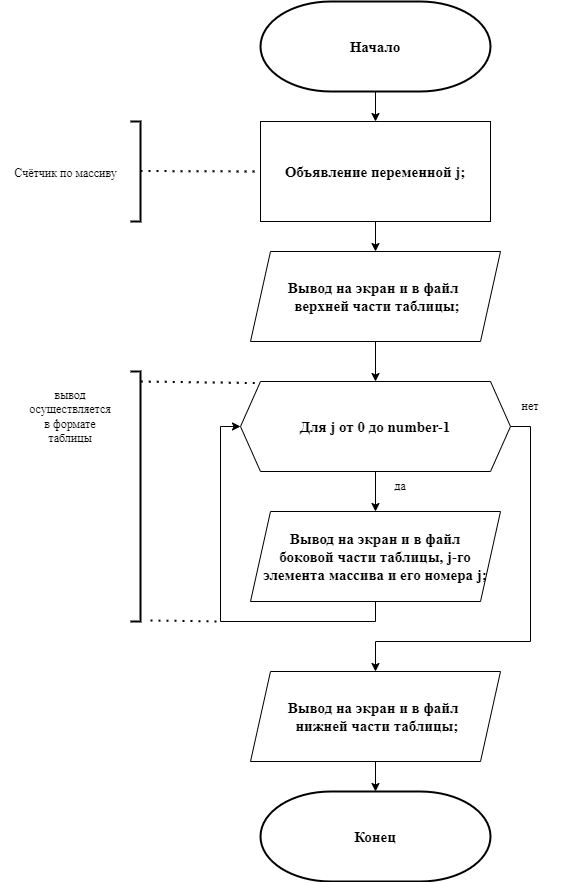
// динамического массива-вектора

, int number) // количество элементов в массиве

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| pechat\_tabl | void | Печать сформированного массива в таблице | Выходной |
| mass\_raspr | double\* | Указатель на первый элемент динамического массива-вектора | Входной |
| number | int | Количество элементов в массиве-векторе | Входной |

**4. Блок-схема функции pechat\_tabl():**



**Описание функции : pechat\_Matr**

**1.Назначение**

Вывод полученной матрицы для гистограмм

**2.Прототип функции**

void pechat\_Matr(double\*\* ppMatr // указатель первого элемента матрицы

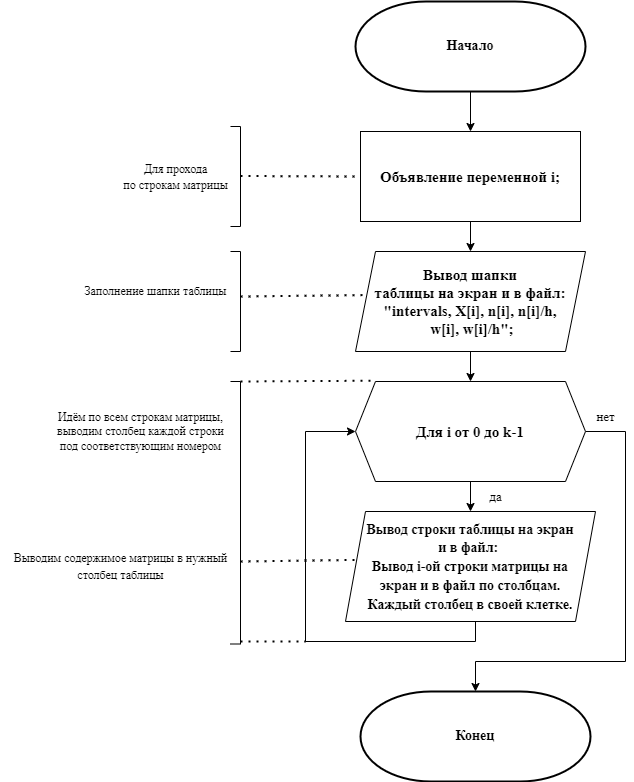
// указателей

, int k) // количество строк в матрице

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| pechat\_ Matr | void | Печать сформированной матрицы в таблице | Выходной |
| ppMatr | double\*\* | Указатель первого элемента матрицы указателей | Входной |
| k | int | Количество строк в матрице для | Входной |

**4. Блок-схема функции pechat\_Matr():**



**Описание функции : delete\_Matr**

**1.Назначение**

Удаление динамической матрицы

**2.** **Прототип функции**

void delete\_Matr(double\*\* ppMatr // указатель на первый элемент матрицы

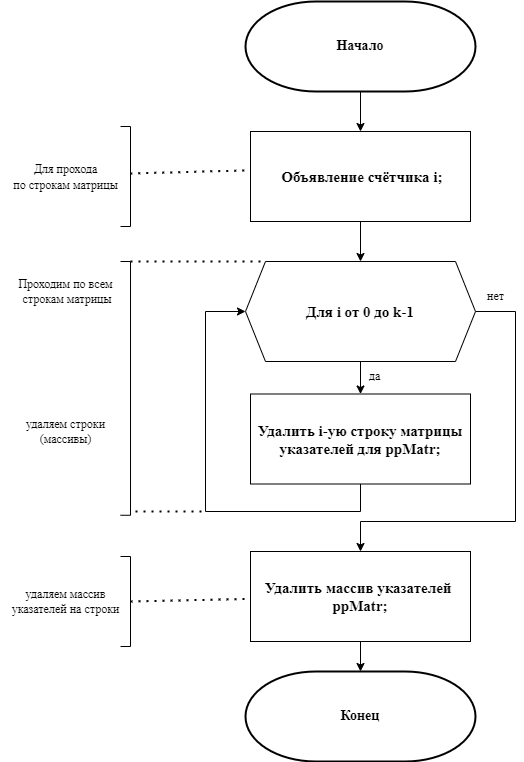
// указателей

, int k) // количество строк в матрице

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| Delete\_Matr | void | Удаление динамической матрицы | Выходной |
| ppMatr | double \*\* | Указатель первого элемента матрицы указателей | Входной |
| k | int | Количество строк матрицы | Входной |

**4. Блок-схема функции delete\_Matr():**



**Описание функции : Box\_Muller\_preobr**

**1.Назначение**

Расчёт нормального распределения преобразованием

**2.** **Прототип функции**

void Box\_Muller\_preobr(double\* mass\_raspr // указатель на первый элемент

// массива преобразования

, double\* A // указатель на первый элемент

// динамического массива

//равномерно распределённой СВ

, double\* B // указатель на первый элемент

// динамического массива

//равномерно распределённой СВ

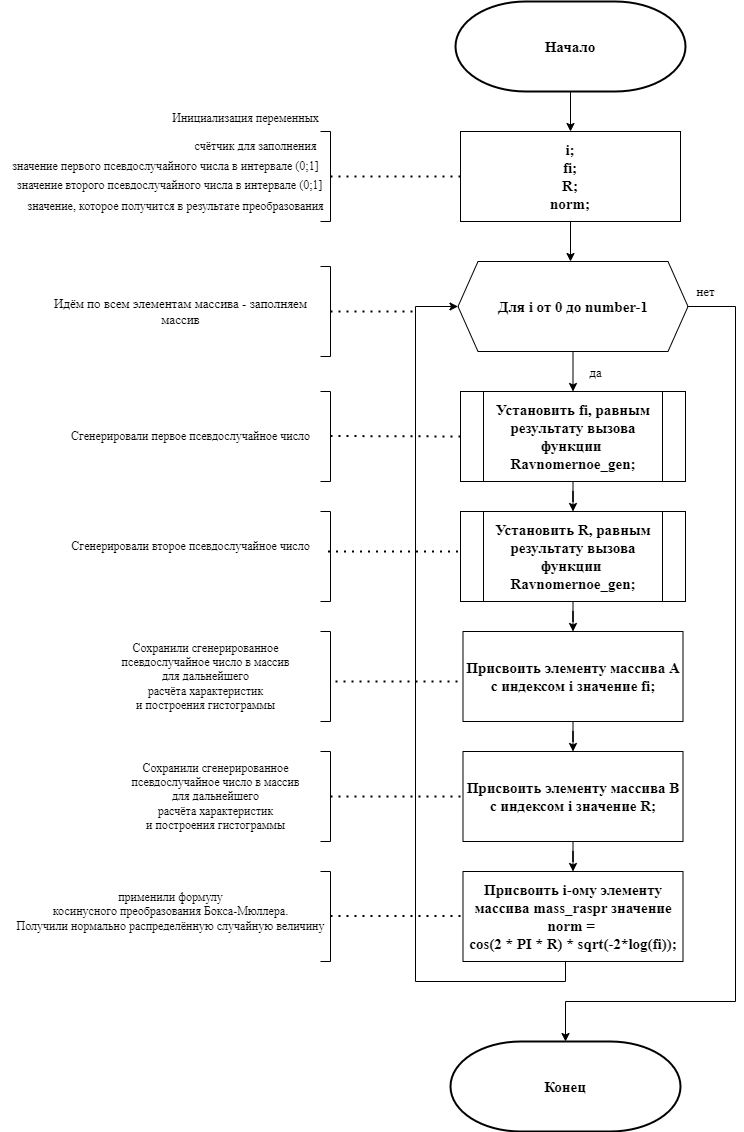
, int number) // количество элементов, которое /

// необходимо сформировать

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| Box\_Muller\_preobr | void | Применение косинусного преобразования Бокса-Мюллера для расчёта нормально распределённой случайной величины | Выходной |
| mass\_raspr | double\* | указатель на первый элемент массива преобразования | Выходной |
| A | double\* | Указатель на первый элемент динамического массива равномерно распределённой СВ | Входной |
| B | double\* | Указатель на первый элемент динамического массива равномерно распределённой СВ | Входной |
| number | int | Количество элементов в массивах | Входной |

**4. Блок-схема функции Box\_Muller\_preobr():**



**Описание функции : Ravnomernoe\_gen**

**1.Назначение**

Создание равномерно распределённых псевдослучайных чисел

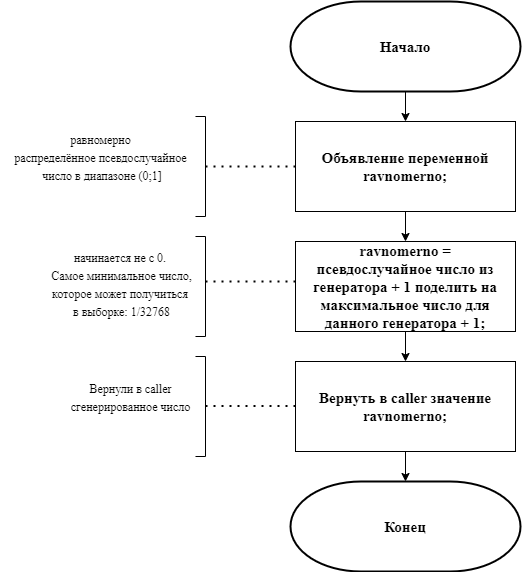
**2.** **Прототип функции**

double Ravnomernoe\_gen(void) // функция ничего не принимает

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| Ravnomernoe\_gen | double | Создание равномерно распределённых псевдослучайных чисел в диапазоне (0;1] | Выходной |

**4. Блок-схема функции Ravnomernoe\_gen():**



**Описание функции : Srednee\_znach**

**1.Назначение**

Подсчёт выборочного среднего

**2.Прототип функции**

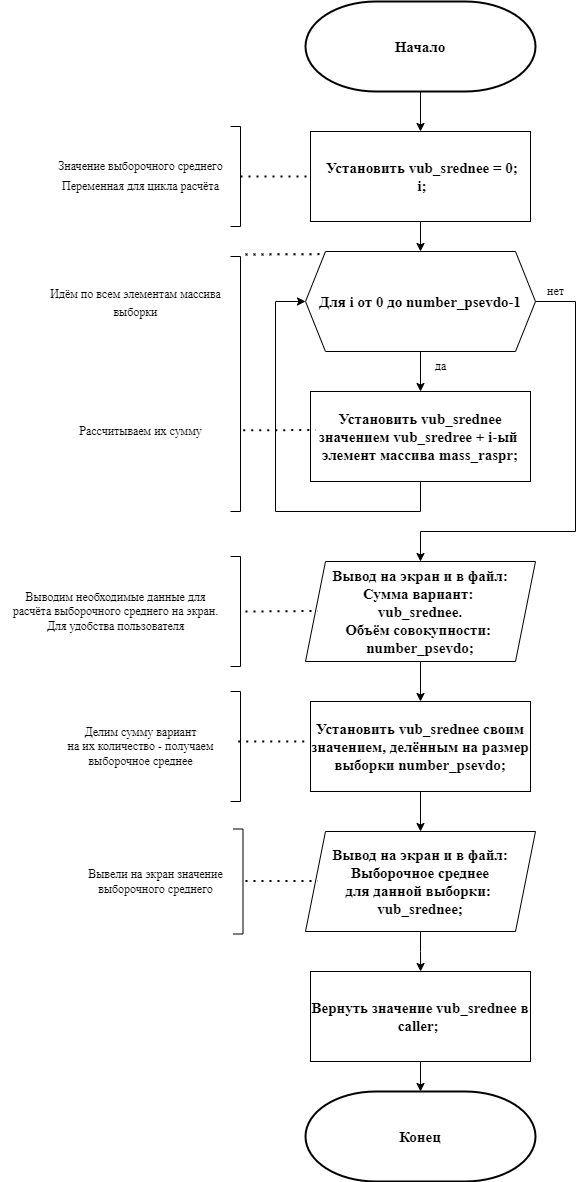
double Srednee\_znach(double\* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

, int number\_psevdo) // количество элементов в массиве

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| Srednee\_znach | double | Подсчёт выборочного среднего в сформированных выборках распределений | Выходной |
| mass\_raspr | double\* | Указатель на первый элемент динамического массива-вектора | Входной |
| number\_psevdo | int | Количество элементов в массиве-векторе | Входной |

**4. Блок-схема функции Srednee\_znach():**

**Описание функции : vub\_dispers**

**1.Назначение**

Подсчёт выборочной дисперсии

**2.Прототип функции**

double vub\_dispers(double\* mass\_raspr // указатель на первый элемент . . . . . . . . . // динамического массива-вектора

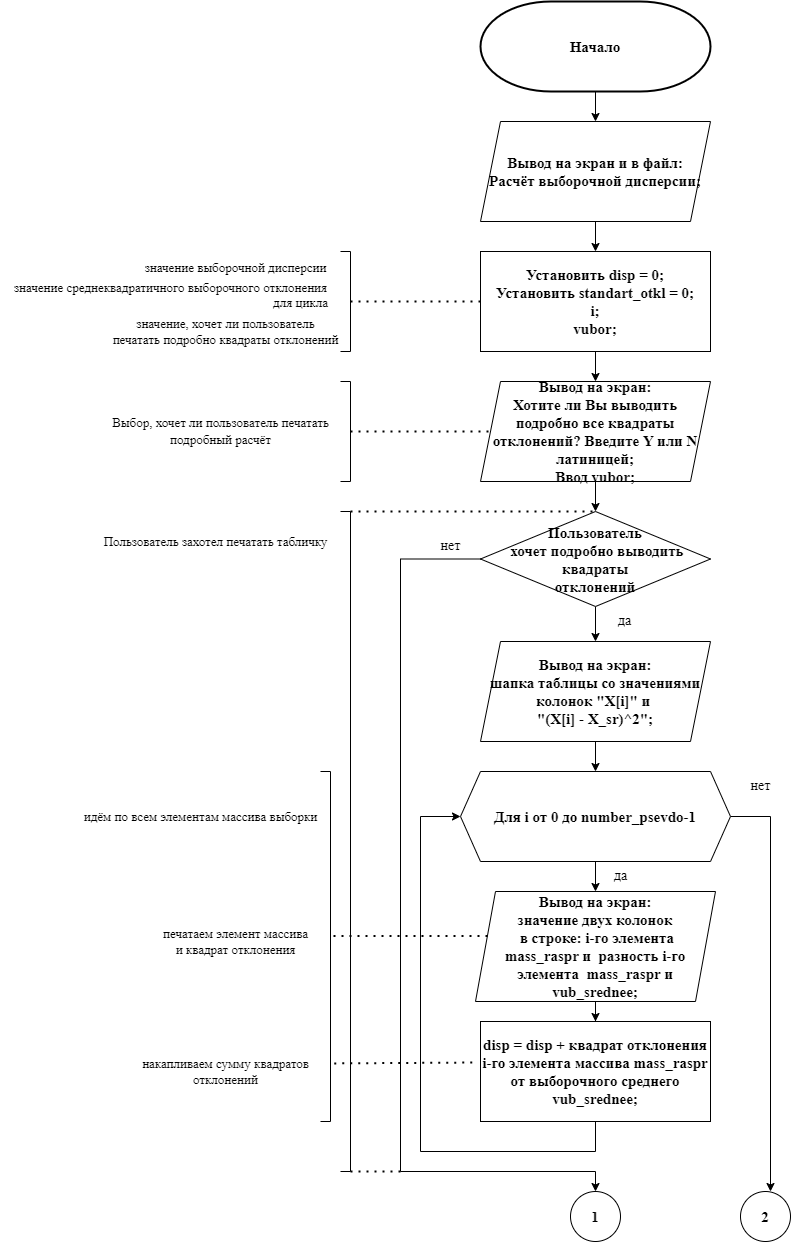
, int number\_psevdo // количество элементов в массиве-векторе

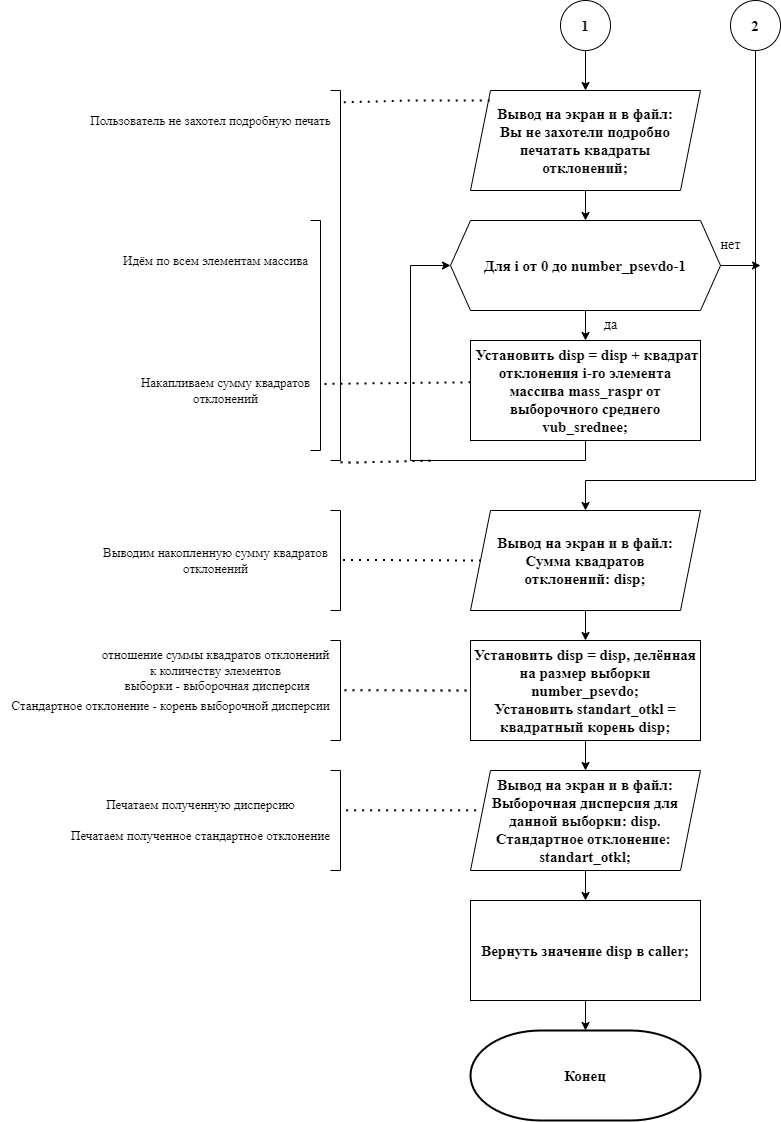
, double vub\_srednee) // рассчитанное ранее выборочное среднее

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| vub\_dispers | double | Подсчёт выборочной дисперсии в сформированных выборках | Выходной |
| number\_psevdo | int | Количество элементов в сформированном массиве выборки | Входной |
| vub\_srednee | double | Рассчитанное выборочное среднее | Входной |

**4. Блок-схема функции vub\_dispers():**





**Описание функции : mediana**

**1.Назначение**

Подсчёт медианы выборки

**2.Прототип функции**

void mediana(double\* mass\_raspr // указатель на первый элемент

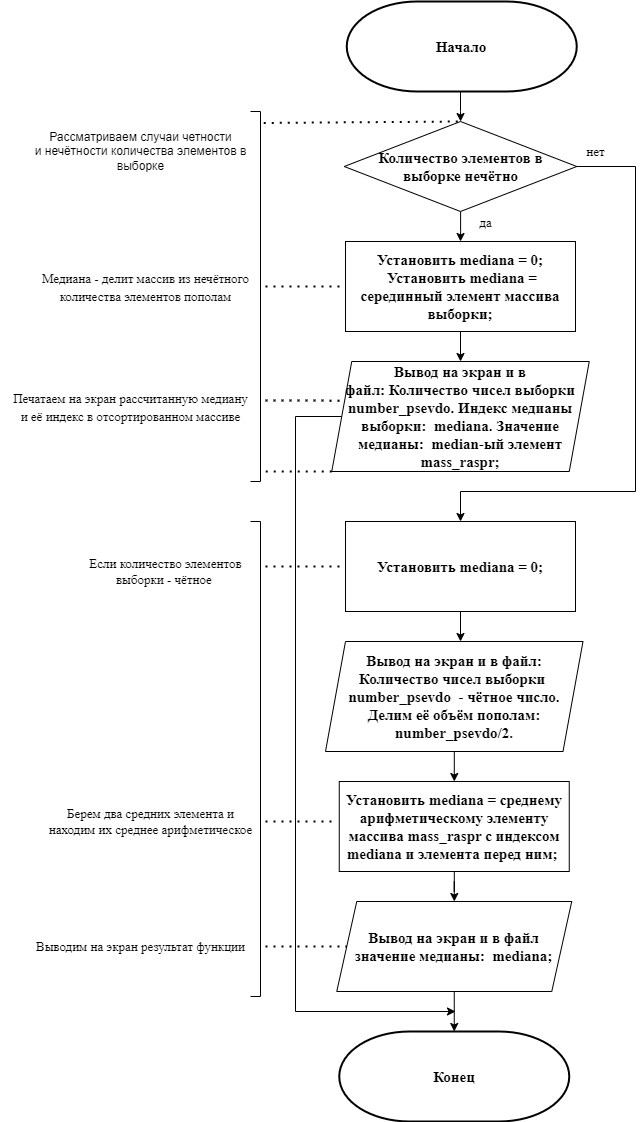
// динамического массива-вектора

, int number\_psevdo) // количество элементов в массиве

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| mediana | void | Подсчёт медианы полученной выборки | Выходной |
| mass\_raspr | double\* | Указатель на первый элемент динамического массива-вектора | Входной |
| number\_psevdo | int | Количество элементов в массиве-векторе | Входной |

**4. Блок-схема функции mediana():**



**Описание функции : gistogramma**

**1.Назначение**

Расчёт интервалов и величин гистограммы

**2.Прототип функции**

void gistogramma(double\* ppMatrA[] // указатель первого элемента матрицы

// указателей

, int& k // количество строк матрицы

, double\* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

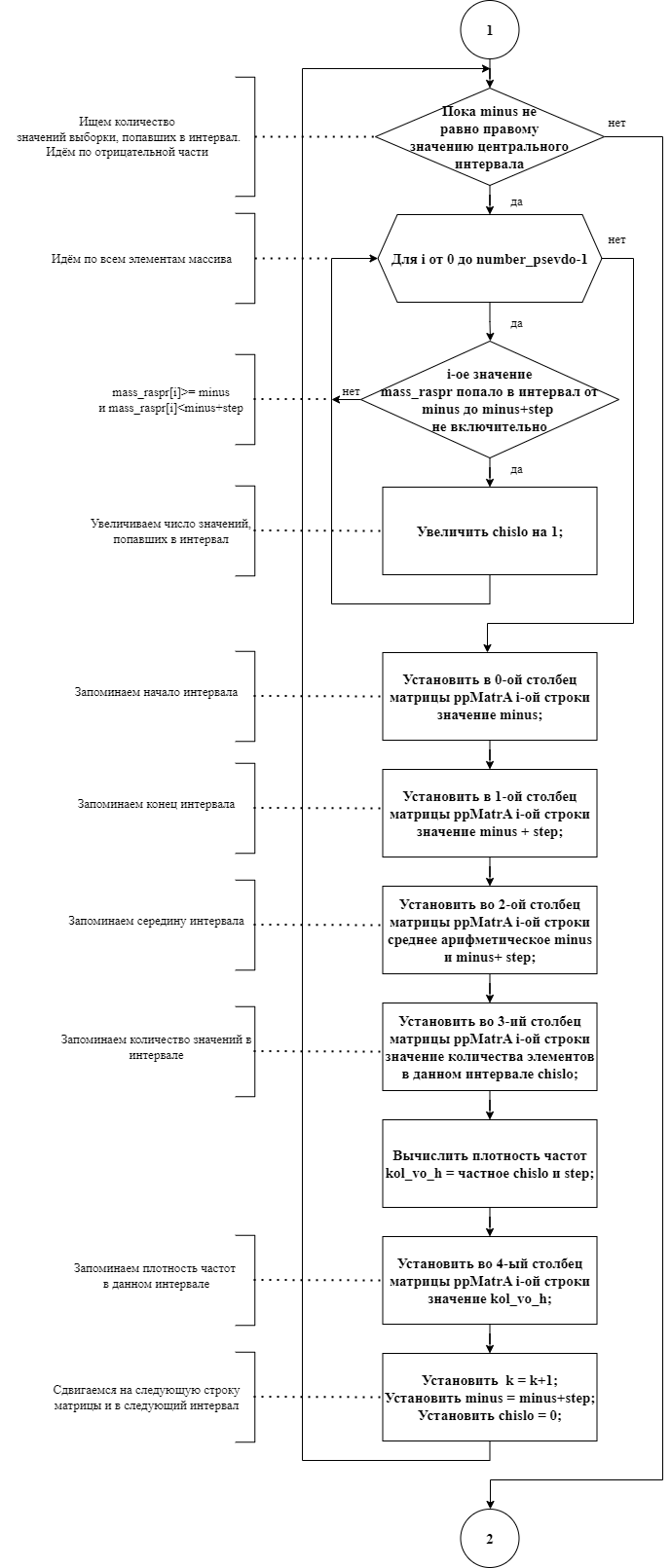
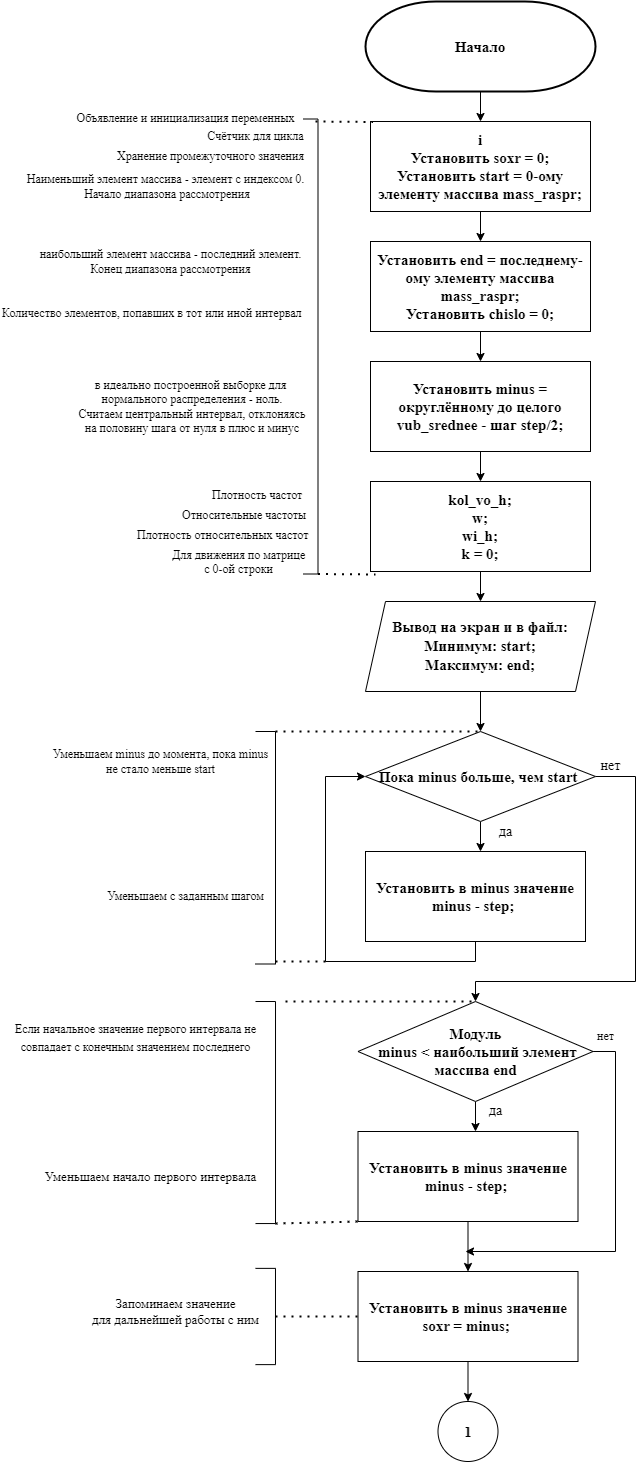
, int number\_psevdo // количество элементов в массиве

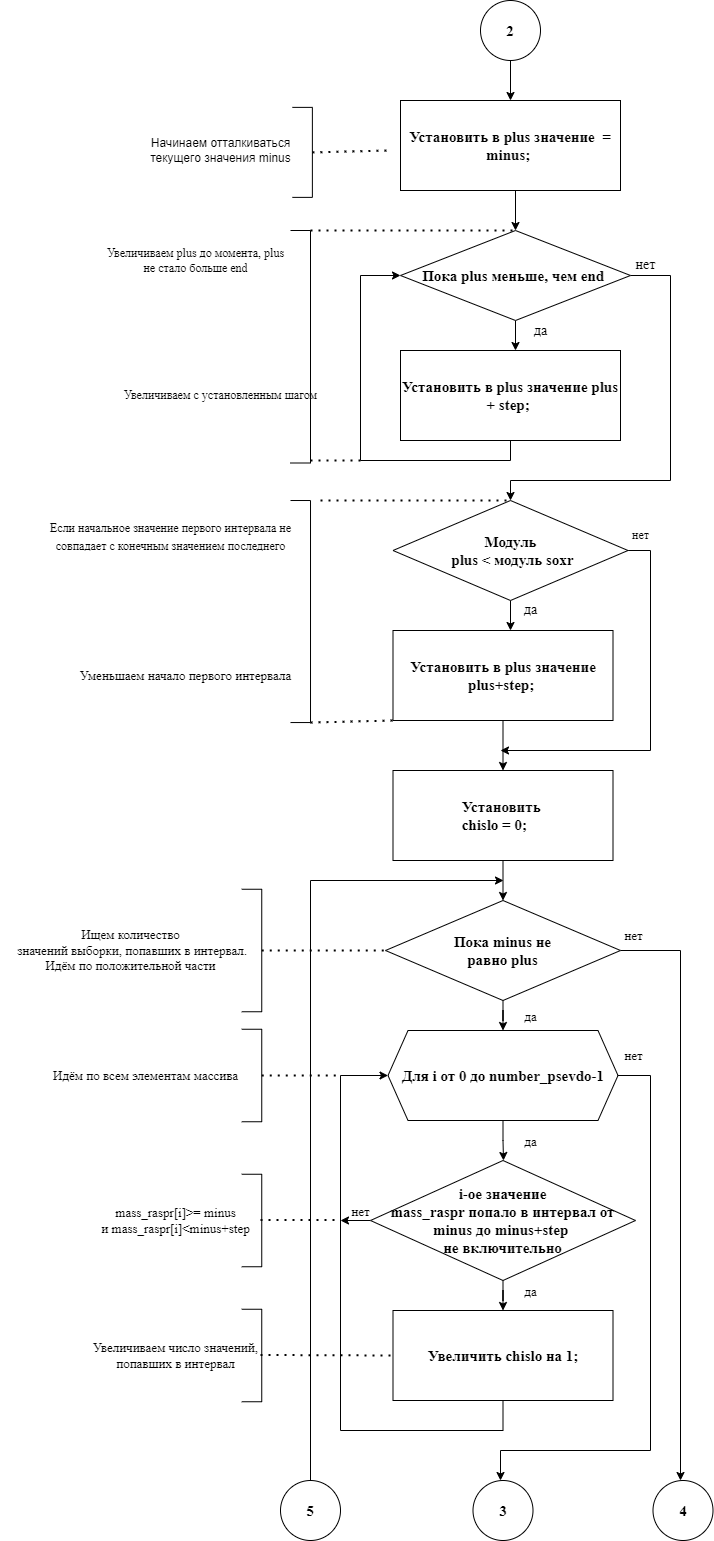
, double vub\_srednee) // рассчитанное выборочное среднее

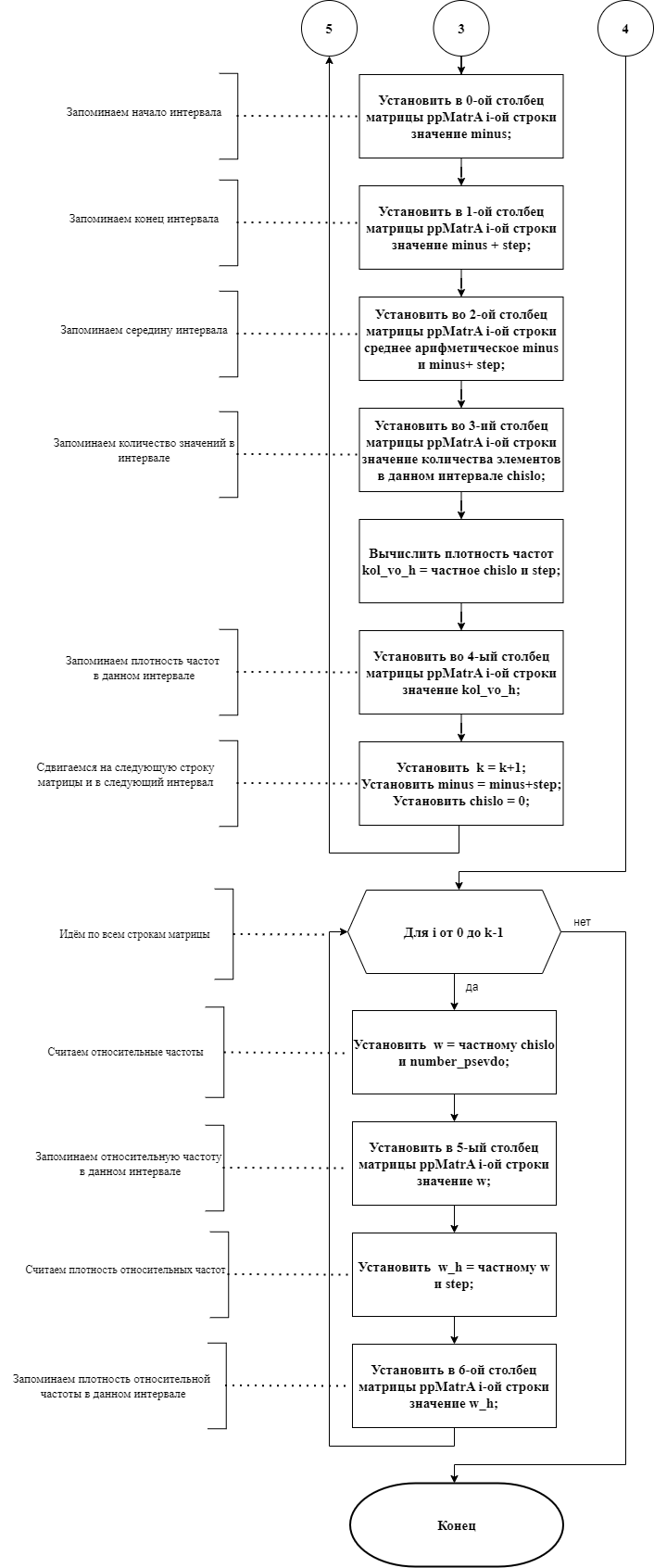
**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| gistogramma | void | Расчёт интервалов и величин гистограммы | Выходной |
| ppMatrA | int\*\* | Указатель первого элемента матрицы указателей | Входной |
| number\_psevdo | int | Количество элементов в сформированном массиве выборки | Входной |
| k | int& | Количество строк матрицы для построения гистограммы для изменения | Входной |
| mass\_raspr | double\* | указатель на первый элемент динамического массива-вектора | Входной |
| vub\_srednee | double | Выборочное среднее нормального распределения | Входной |

**4. Блок-схема функции gistogramma():**







**Описание функции : gistogramma\_ravnomerno**

**1.Назначение**

Расчёт интервалов и величин гистограммы для равномерного распределения

**2.Прототип функции**

void gistogramma\_ravnomerno(double\* ppMatrA[] // указатель первого

// элемента матрицы указателей

, int& k // количество строк матрицы

, double\* mass\_raspr // указатель на первый элемент

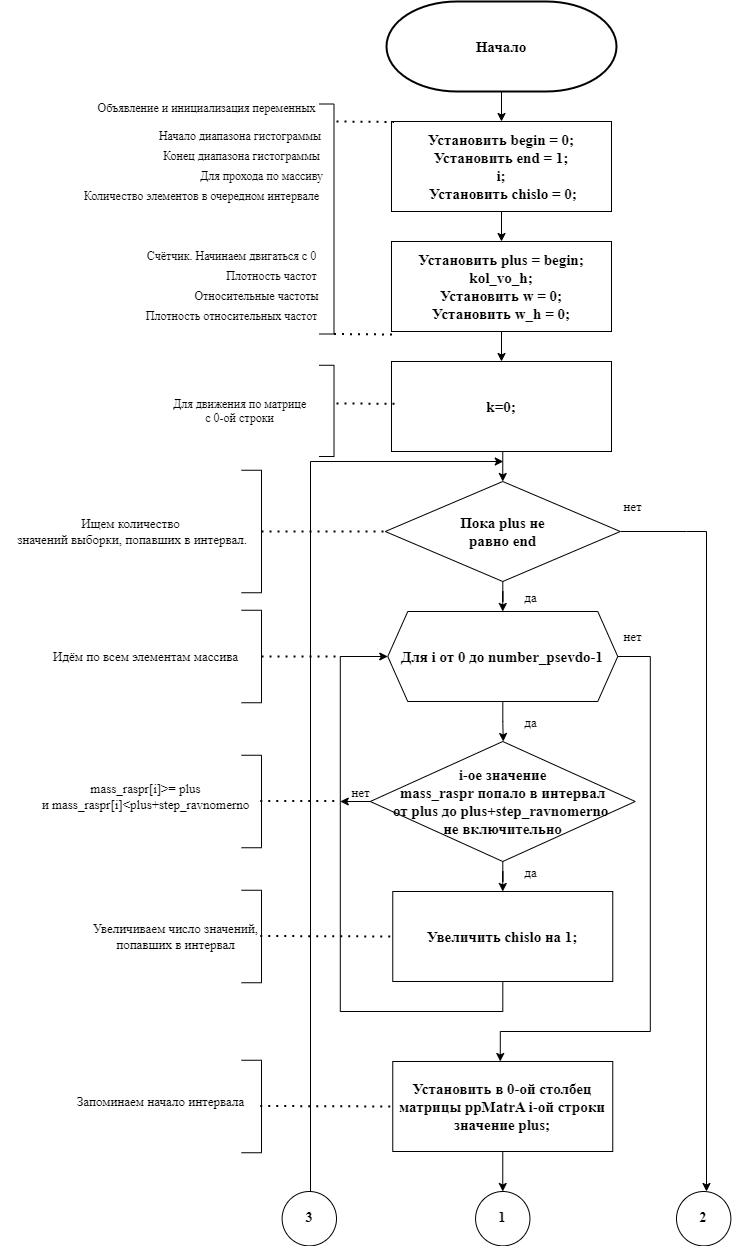
// динамического массива-вектора

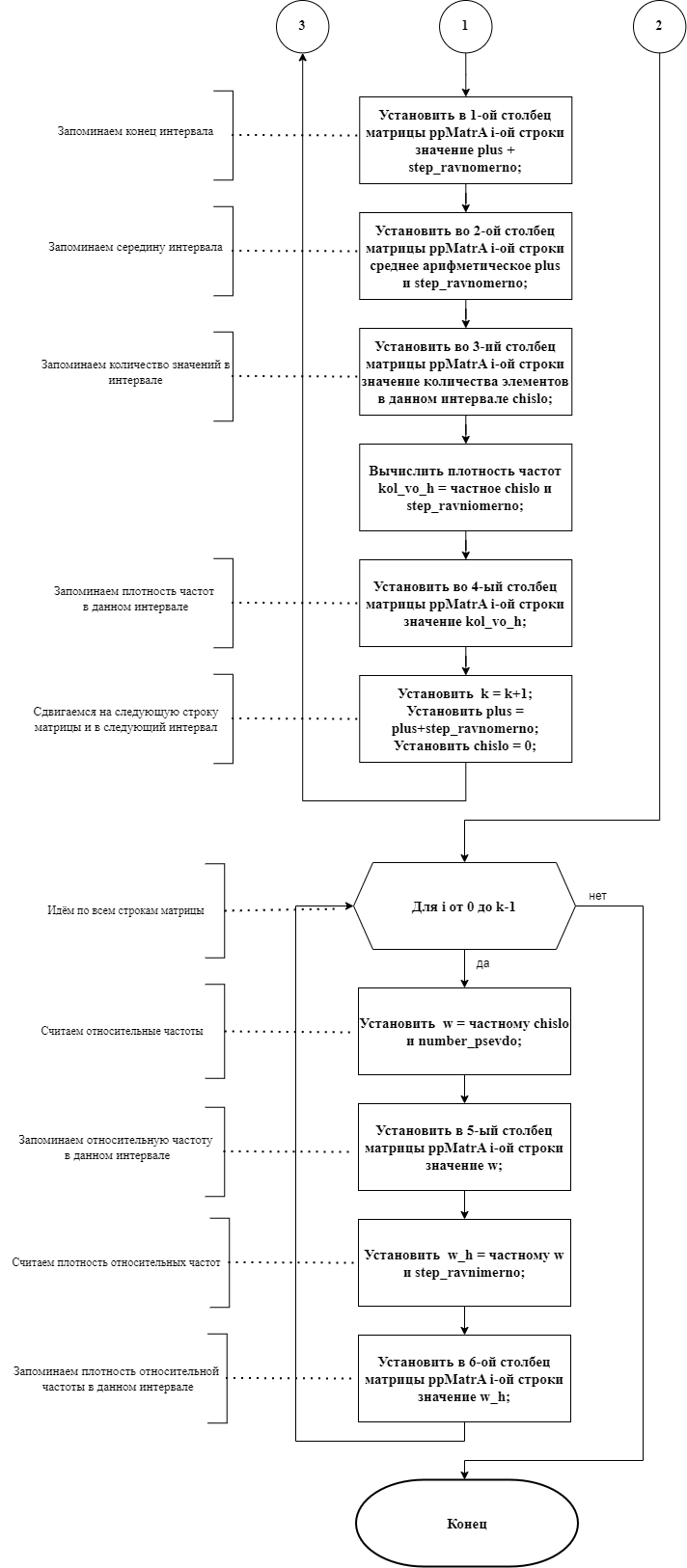
, int number\_psevdo) // количество элементов в массиве

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| gistogramma\_ravnomerno | void | Расчёт интервалов и величин гистограммы равномерного распределения | Выходной |
| ppMatrA | int\*\* | Указатель первого элемента матрицы указателей | Входной |
| number\_psevdo | int | Количество элементов в сформированном массиве выборки | Входной |
| k | int& | Количество строк матрицы для построения гистограммы для изменения | Входной |
| mass\_raspr | double\* | указатель на первый элемент динамического массива-вектора | Входной |

**4. Блок-схема функции gistogramma\_ravnomerno():**





**Описание функции : moda**

**1.Назначение**

Подсчёт моды выборки

**2.Прототип функции**

void moda(double\* ppMatrA[] // указатель первого элемента матрицы

// указателей

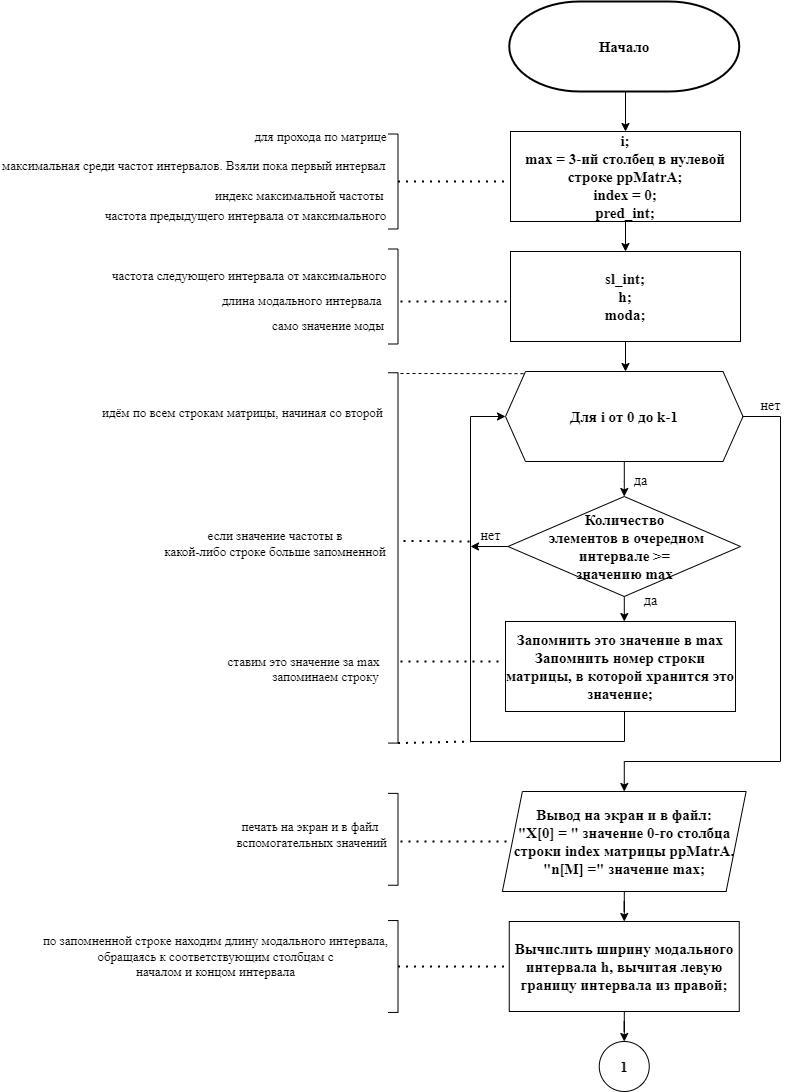
, int k // количество строк матрицы

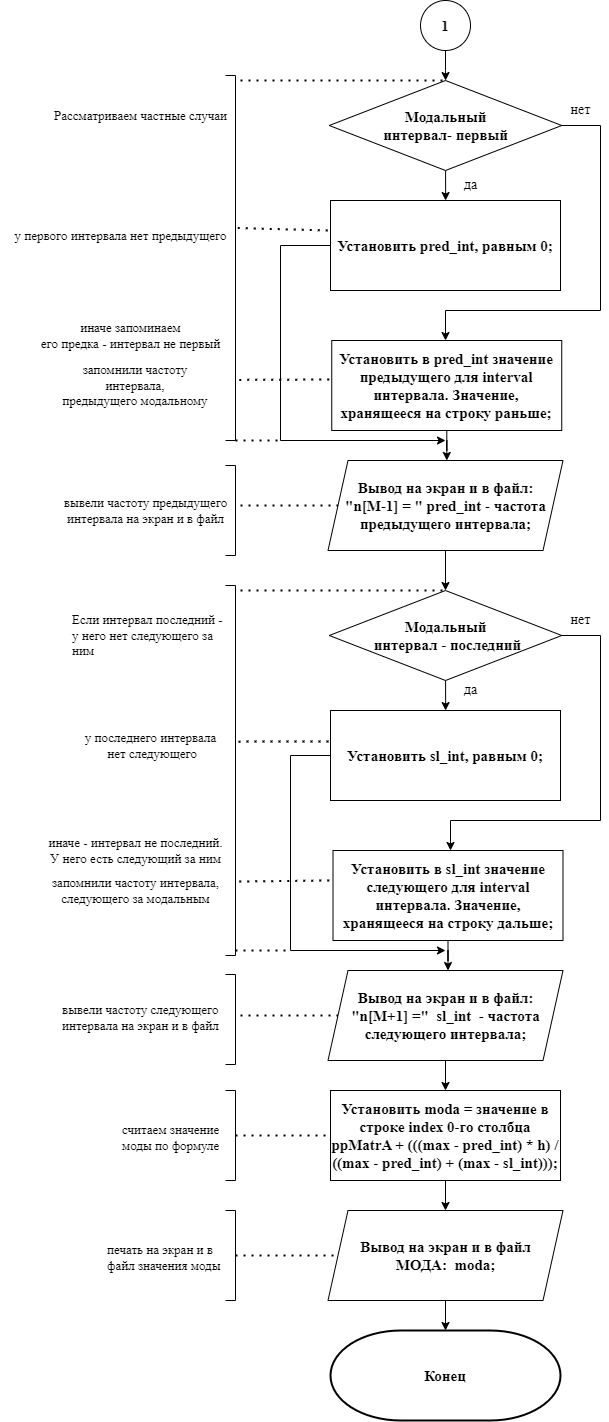
, int columns) // количество столбцов матрицы

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| moda | void | Подсчёт моды выборки | Выходной |
| ppMatrA | int\*\* | Указатель первого элемента матрицы указателей | Входной |
| k | int | Количество строк матрицы для гистограммы | Входной |
| columns | int | Количество столбцов в матрице гистограммы | Входной |

**4. Блок-схема функции moda():**





**Описание функции : Merge\_sort\_vozr**

**1.Назначение**

Сортировка слиянием по возрастанию

**2.Прототип функции**

void Merge\_sort\_vozr(double\* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

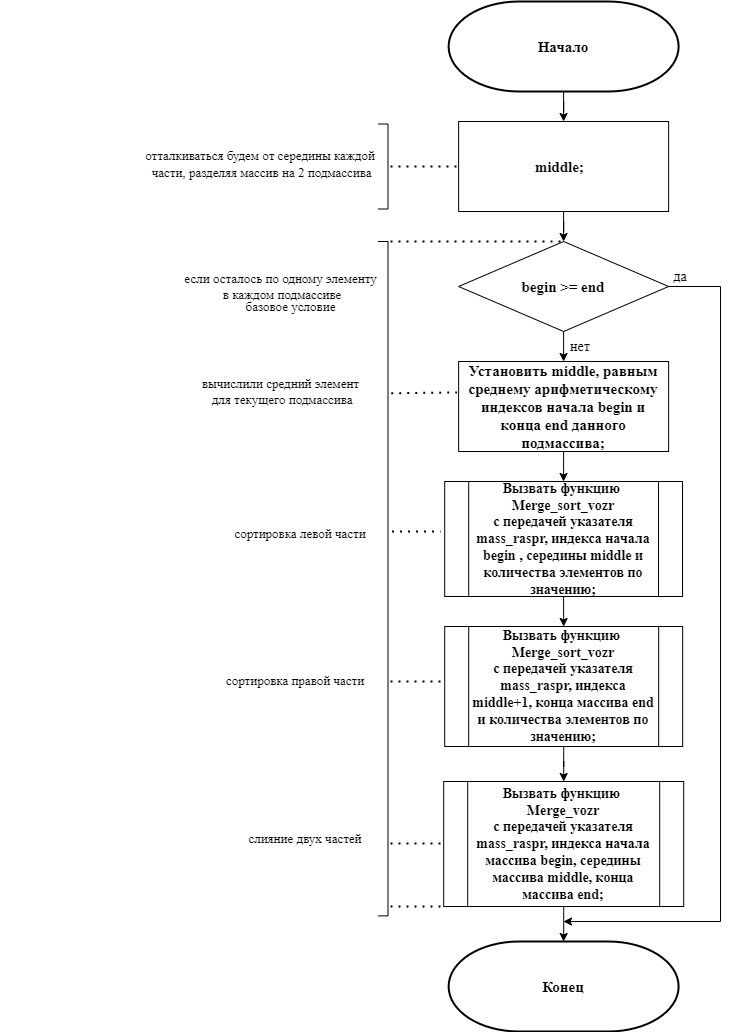
, int begin // начало динамического массива

, int end) // конец динамического массива

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| Merge\_sort\_vozr | void | Сортировка слиянием по возрастанию | Выходной |
| mass\_raspr | double\* | Указатель на первый элемент динамического массива-вектора | Входной |
| begin | int | Начало динамического массива-вектора | Входной |
| end | int | Конец динамического массива-вектора | Входной |

**4. Блок-схема функции Merge\_sort\_vozr():**



**Описание функции : Merge\_vozr**

**1.Назначение**

"Властвование" - сливание массивов для Merge\_sort\_vozr

**2.Прототип функции**

void Merge\_vozr(double\* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

, int begin // начало динамического массива

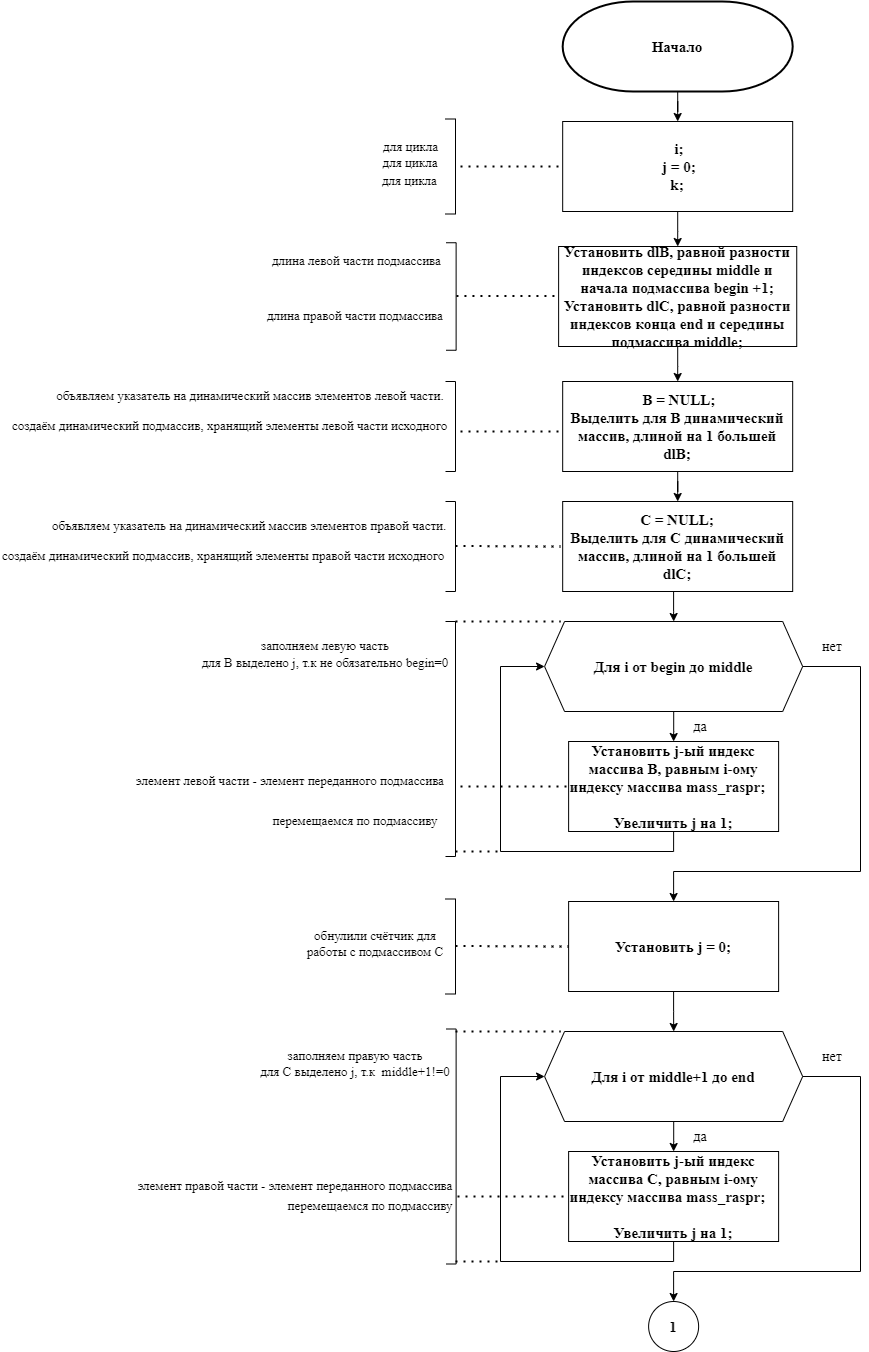
, int middle // середина динамического массива

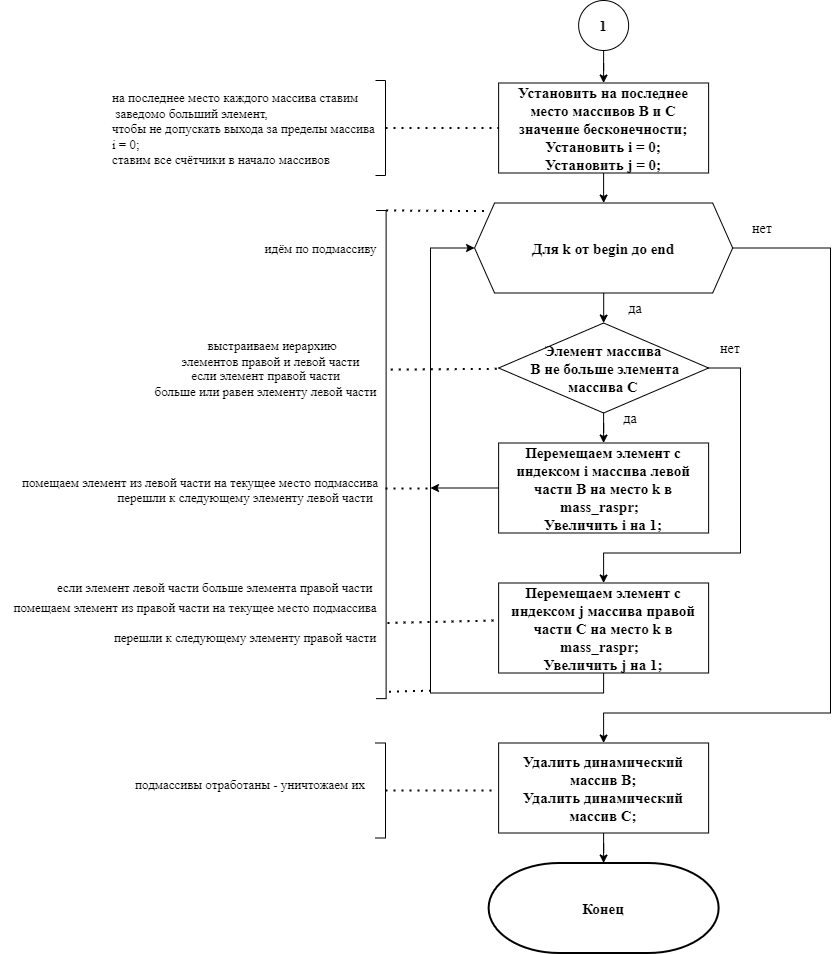
, int end) // конец динамического массива

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| Merge\_vozr | void | "Властвование" - сливание массивов для Merge\_sort\_vozr | Выходной |
| mass\_raspr | double\* | Указатель на первый элемент динамического массива-вектора | Входной |
| begin | int | Начало динамического массива-вектора | Входной |
| middle | int | Середина динамического массива-вектора | Входной |
| end | int | Конец динамического массива-вектора | Входной |

**4. Блок-схема функции Merge\_vozr():**





**Описание функции : teor\_plotnost\_norm**

**1.Назначение**

Расчёт теоретической плотности нормального распределения

**2.Прототип функции**

void teor\_plotnost\_norm(double vub\_srednee // рассчитанное выборочное

// среднее

, double vub\_disp // рассчитанная выборочная

// дисперсия

, int number\_psevdo // количество элементов в

// выборке

, double\* ppMatrA[] // указатель первого элемента

// матрицы указателей

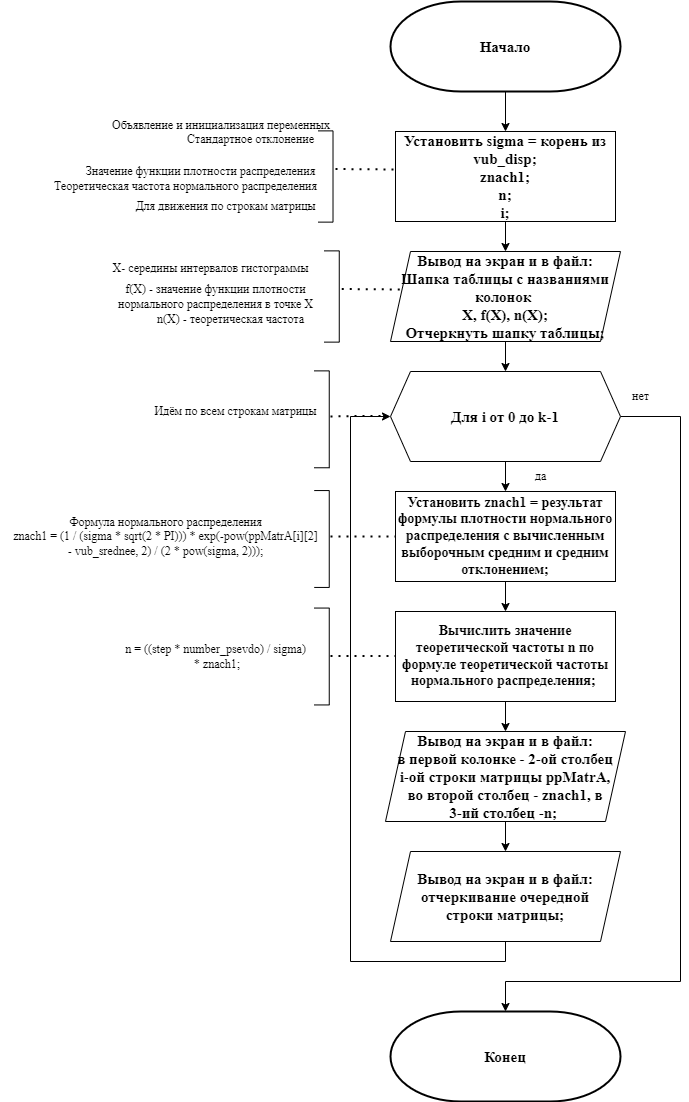
, int k) // количество строк в матрице

// указателей

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| teor\_plotnost\_norm | void | Расчёт теоретической плотности нормального распределения | Выходной |
| ppMatrA | int\*\* | Указатель первого элемента матрицы указателей | Входной |
| number\_psevdo | int | Количество элементов в сформированном массиве выборки | Входной |
| k | int | Количество строк в матрице гистограммы | Входной |
| vub\_disp | double | Рассчитанная выборочная дисперсия | Входной |
| vub\_srednee | double | Рассчитанное выборочное среднее | Входной |

**4. Блок-схема функции teor\_plotnost\_norm():**



**Описание функции : teor\_plotnost\_ravnomerno**

**1.Назначение**

Расчёт теоретической плотности равномерного распределения

**2.Прототип функции**

void teor\_plotnost\_ravn(double vub\_disp // рассчитанная выборочная

// дисперсия

, int number\_psevdo // количество элементов в выборке

, double\* ppMatrA[] // указатель первого элемента

// матрицы указателей

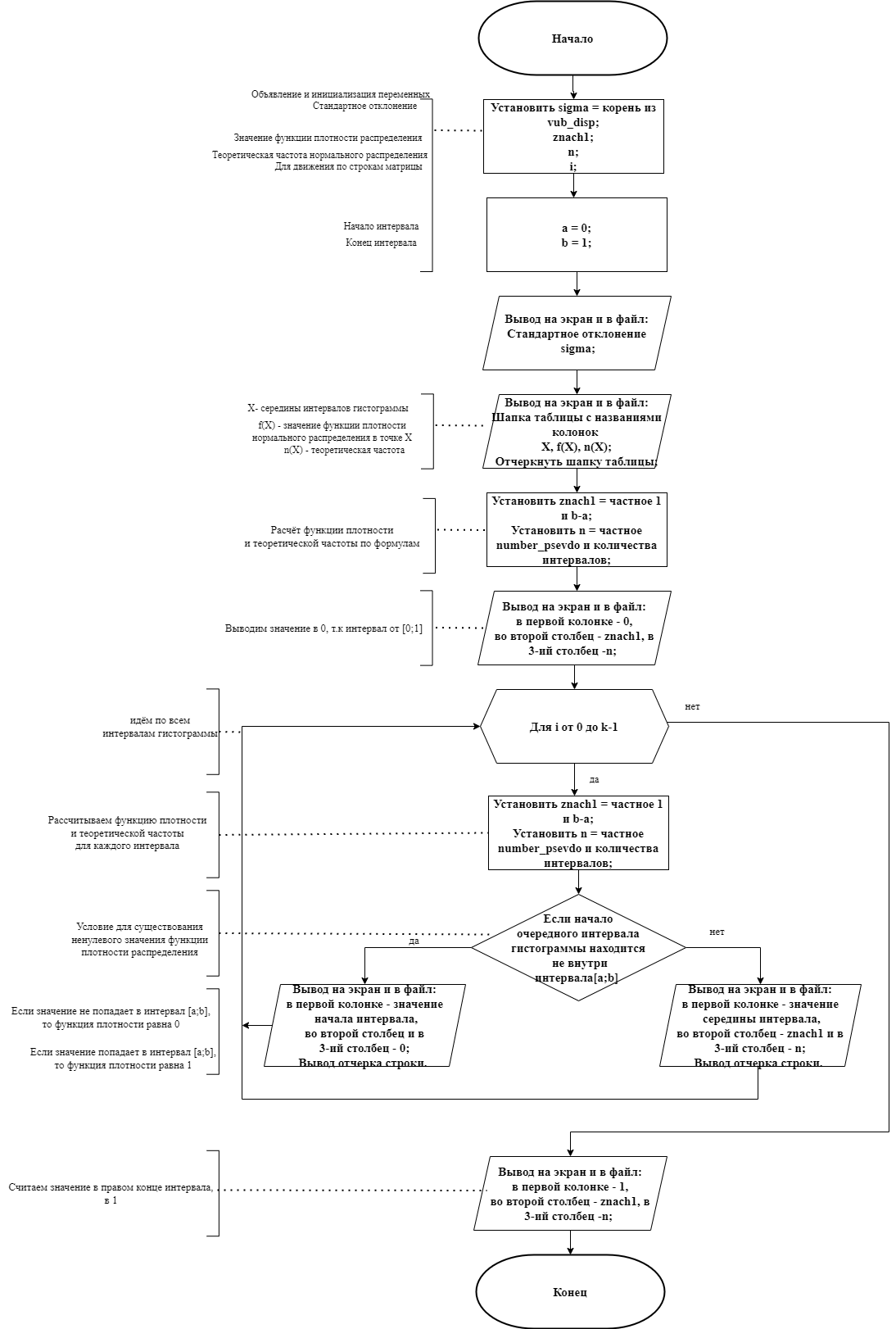
, int k) // количество строк в матрице

// указателей

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| teor\_plotnost\_ravn | void | Расчёт теоретической плотности равномерного распределения | Выходной |
| ppMatrA | int\*\* | Указатель первого элемента матрицы указателей | Входной |
| number\_psevdo | int | Количество элементов в сформированном массиве выборки | Входной |
| k | int | Количество строк в матрице гистограммы | Входной |
| vub\_disp | double | Рассчитанная выборочная дисперсия | Входной |

**4. Блок-схема функции teor\_plotnost\_ravnomerno():**



**Описание функции : pract\_raspr**

**1.Назначение**

Расчёт практического псевдослучайного процесса для нормального и равномерного распределения

**2.Прототип функции**

void pract\_raspr(double\* ppMatrA[] // указатель первого элемента матрицы

// указателей

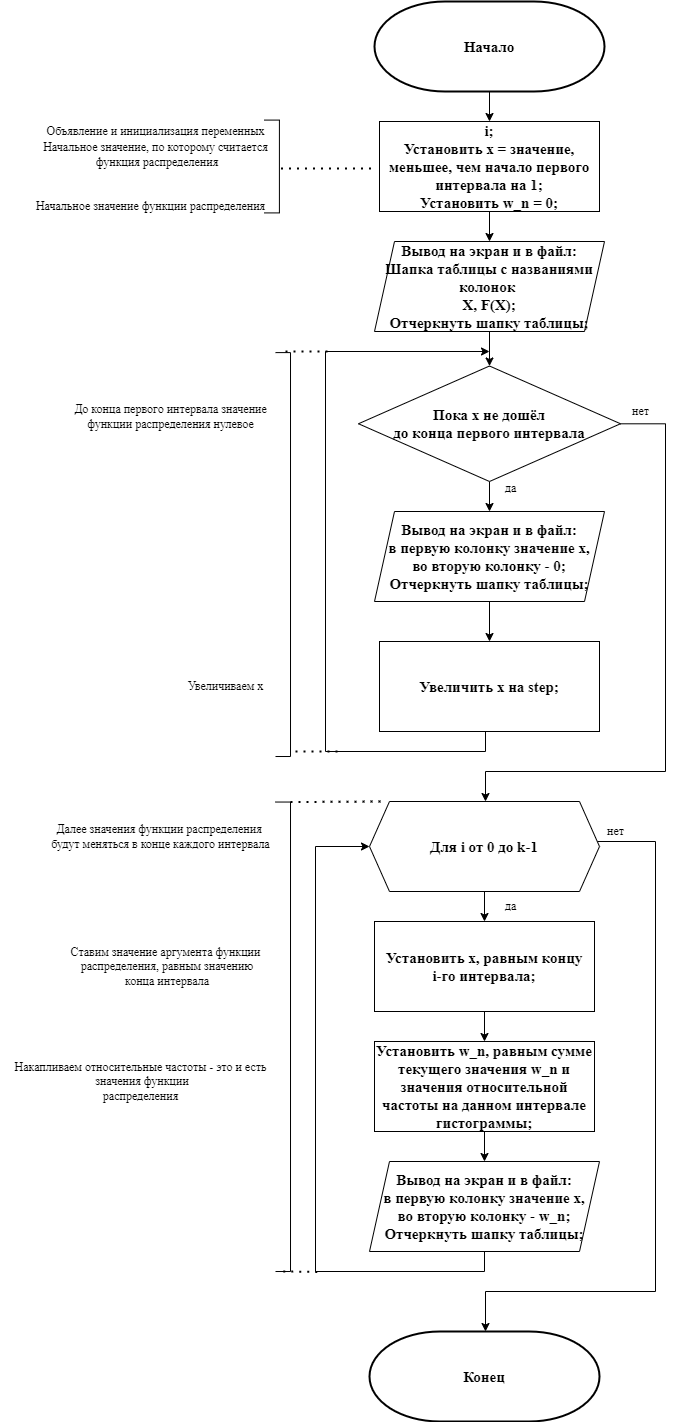
, int k) // количество строк в матрице

// указателей

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| pract\_raspr | void | Расчёт практического псевдослучайного процесса для нормального и равномерного распределения | Выходной |
| ppMatrA | int\*\* | Указатель первого элемента матрицы указателей | Входной |
| k | int | Количество строк в матрице гистограммы | Входной |

**4. Блок-схема функции pract\_raspr():**



**Описание функции : teor\_raspr\_norm**

**1.Назначение**

Расчёт теоретического псевдослучайного процесса для нормального распределения

**2.Прототип функции**

void teor\_raspr\_norm(double vub\_srednee // рассчитанное выборочное среднее

, double vub\_disp // рассчитанная выборочная

// дисперсия

, double\* ppMatrA[] // указатель первого элемента

// матрицы указателей

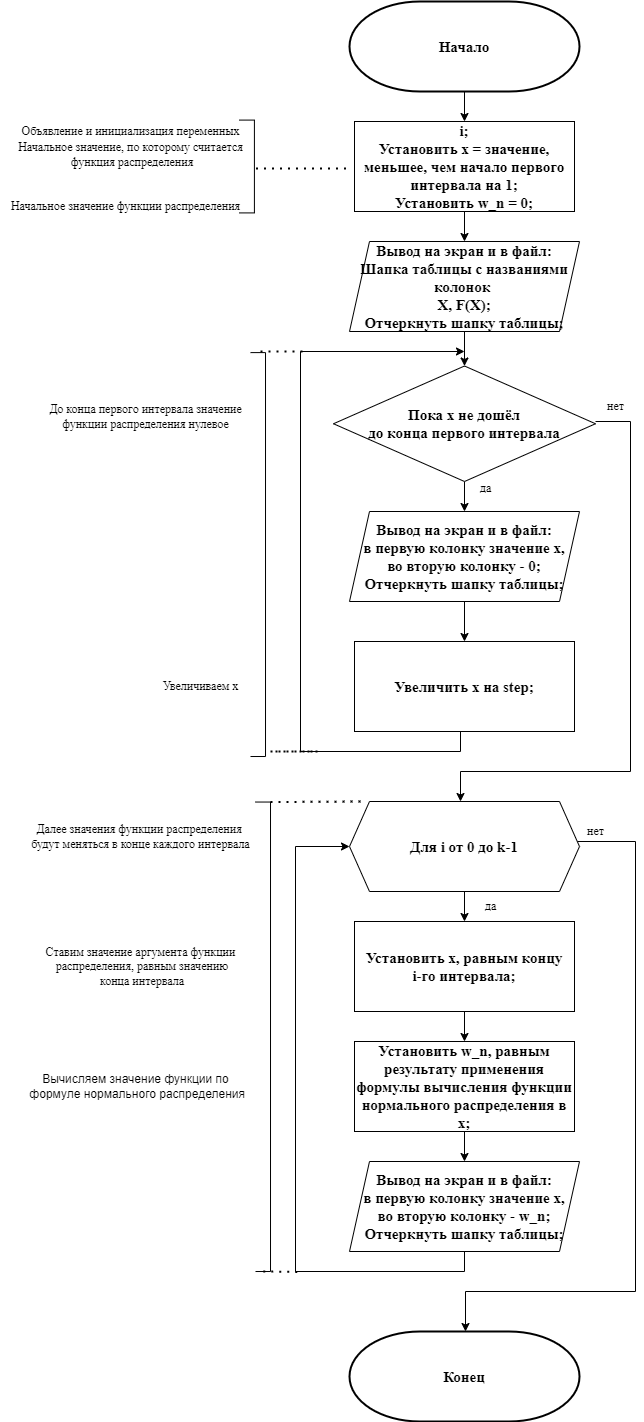
, int k) // количество строк в матрице

// указателей

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| teor\_raspr\_norm | void | Расчёт теоретического псевдослучайного процесса для нормального распределения | Выходной |
| ppMatrA | int\*\* | Указатель первого элемента матрицы указателей | Входной |
| vub\_disp | double | Рассчитанная выборочная дисперсия данного распределения | Входной |
| vub\_srednee | double | Рассчитанное выборочное среднее данного распределения | Входной |
| k | int | Количество строк в матрице гистограммы | Входной |

**4. Блок-схема функции teor\_raspr\_norm():**



**Описание функции : teor\_raspr\_ravnom**

**1.Назначение**

Расчёт теоретического псевдослучайного процесса для равномерного распределения

**2.Прототип функции**

void teor\_raspr\_ravnom(double\* ppMatrA[] // указатель первого элемента

// матрицы указателей

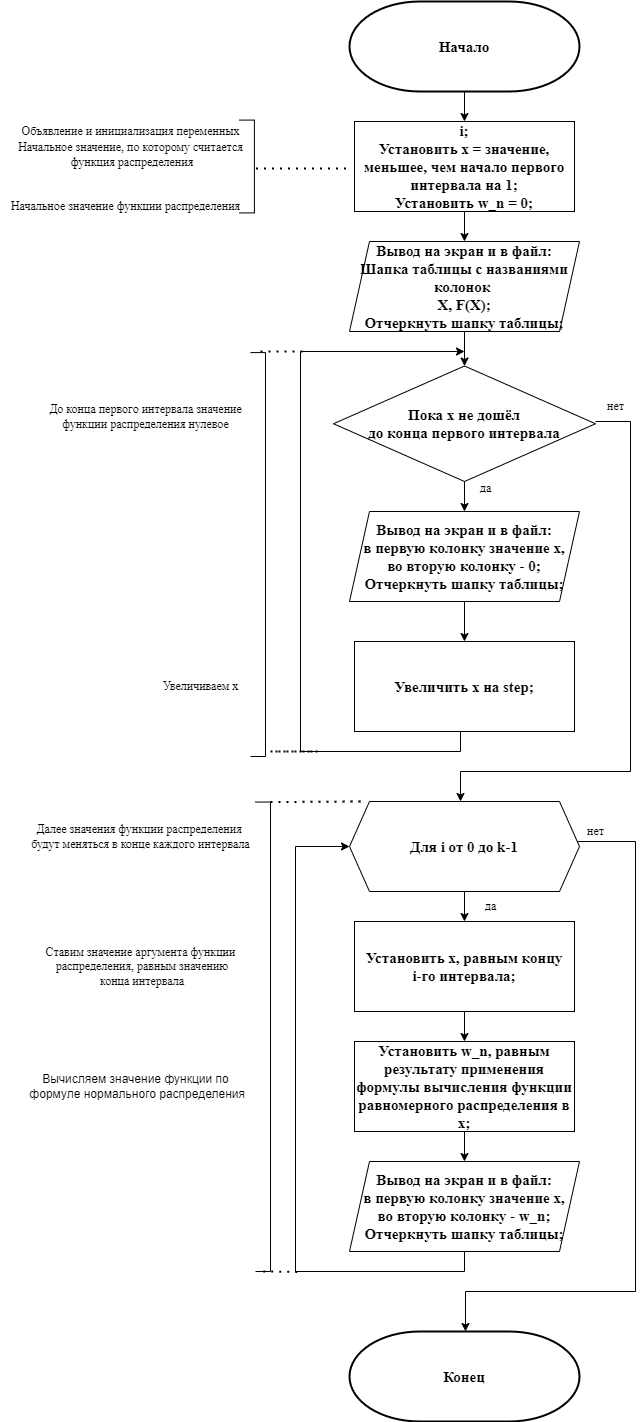
, int k) // количество строк в матрице

// указателей

**3. Описание параметров**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип | Назначение | Входной / Выходной |
| teor\_raspr\_ravnom | void | Расчёт теоретического псевдослучайного процесса для равномерного распределения | Выходной |
| ppMatrA | int\*\* | Указатель первого элемента матрицы указателей | Входной |
| k | int | Количество строк в матрице гистограммы | Входной |

**4. Блок-схема функции teor\_raspr\_ravnom():**



**Код программы**

#include <iostream> // стандартные потоки ввода/вывода

#include <cmath> // библиотека с математическими функциями

#include <iomanip> // библиотека с модификатором setw

#include <fstream> // файловые потоки ввода/вывода

using namespace std; // используем пространство имён std

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Г Л О Б А Л Ь Н Ы Е К О Н С Т А Н Т Ы \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// глобальная константа числа пи

const double PI = 3.141592653589793;

// шаг разбития гистограммы нормального распределения

const float step = 0.5;

// шаг разбиения для гистограммы равномерных процессов

const float step\_ravnomerno = 0.2;

// путь к файлу через константный указатель

const char\* FNAME = "Result.txt";

const char\* FNAME\_READ = "Number\_of pairs.txt";

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Г Л О Б А Л Ь Н Ы Е Ф А Й Л О В Ы Е П О Т О К И \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// объявление объекта для потокового вывода данных в файл

ofstream fout;

// объявление объекта для потокового ввода данных из файла

ifstream fin;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* П Р О Т О Т И П Ы Ф У Н К Ц И Й \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// удаление динамического массива - вектора

void udalenie(double\* mass\_raspr); // указатель на первый элемент динамического

// массива-вектора

// вывод сформированного массива результатов преобразования

void pechat\_tabl(double\* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

,int number); // количество элементов в массиве-векторе

// вывод полученной матрицы

void pechat\_Matr(double\*\* ppMatr // указатель первого элемента матрицы

// указателей

, int k); // количество строк в матрице

//освободить память матрицы

void delete\_Matr(double\*\* ppMatr // указатель на первый элемент матрицы

// указателей

, int k); // количество строк в матрице

// расчёт нормального распределения преобразованием

void Box\_Muller\_preobr(double\* mass\_raspr // указатель на первый элемент

// массива преобразования

, double\* A // указатель на первый элемент

// динамического массива

// равномерно распределённой СВ

, double\* B // указатель на первый элемент

// динамического массива

// равномерно распределённой СВ

, int number); // количество элементов, которое

// необходимо сформировать

// создание равномерно распределённых псевдослучайных чисел

double Ravnomernoe\_gen(void); // функция ничего не принимает

// подсчёт выборочного среднего

double Srednee\_znach(double\* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

, int number\_psevdo); // количество элементов в массиве

// подсчёт выборочной дисперсии

double vub\_dispers(double\* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

, int number\_psevdo // количество элементов в массиве

, double vub\_srednee); // рассчитанное выборочное среднее

// подсчёт медианы выборки

void mediana(double\* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

, int number\_psevdo); // количество элементов в массиве-векторе

// подсчёт моды выборки

void moda(double\* ppMatrA[] // указатель первого элемента матрицы

// указателей

, int k // количество строк матрицы

, int columns); // количество столбцов матрицы

// расчёт интервалов и величин гистограммы

void gistogramma(double\* ppMatrA[] // указатель первого элемента матрицы

// указателей

, int& k // количество строк матрицы

, double\* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

, int number\_psevdo // количество элементов в массиве

, double vub\_srednee); // расcчитанное выборочное среднее

// расчёт интервалов и величин гистограммы

void gistogramma\_ravnomerno(double\* ppMatrA[] // указатель первого элемента

// матрицы указателей

, int& k // количество строк матрицы

// для изменения

, double\* mass\_raspr // указатель на первый

// элемент динамического

// массива-вектора

, int number\_psevdo); // количество элементов в

// массиве-векторе

// сортировка слиянием по возрастанию

void Merge\_sort\_vozr(double \*mass\_raspr // указатель на первый элемент

// динамического массива-вектора

, int begin // начало динамического массива

, int end); // конец динамического массива

// "Властвование" - сливание массивов для Merge\_sort\_vozr

void Merge\_vozr(double \* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

, int begin // начало динамического массива

, int middle // середина динамического массива

, int end); // конец динамического массива

// расчёт теоретической плотности нормального распределения

void teor\_plotnost\_norm(double vub\_srednee // расcчитанное выборочное среднее

, double vub\_disp // расcчитанная выборочная

// дисперсия

, int number\_psevdo // количество элементов в выборке

, double\* ppMatrA[] // указатель первого элемента

// матрицы указателей

, int k); // количество строк в матрице

// указателей

// расчёт теоретической плотности равномерного распределения

void teor\_plotnost\_ravn(double vub\_disp // расcчитанная выборочная

// дисперсия

, int number\_psevdo // количество элементов в выборке

, double\* ppMatrA[] // указатель первого элемента

// матрицы указателей

, int k); // количество строк в матрице

// указателей

// расчет функции псевдослучайных процессов

void pract\_raspr(double\* ppMatrA[] // указатель первого элемента матрицы

// указателей

, int k); // количество строк в матрице указателей

// расчет теоретической функции гауссовского процесса

void teor\_raspr\_norm(double vub\_srednee // расcчитанное выборочное среднее

, double vub\_disp // расcчитанная выборочная дисперсия

, double\* ppMatrA[] // указатель первого элемента матрицы

// указателей

, int k); // количество строк в матрице

// указателей

// расчет теоретической функции равномерного процесса

void teor\_raspr\_ravnom(double\* ppMatrA[] // указатель первого элемента

// матрицы указателей

, int k); // количество строк в матрице

// указателей

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* О С Н О В Н А Я П Р О Г Р А М М А \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

system("color F0"); // делаем консоль светлой

setlocale(LC\_ALL, "Rus"); // подключаем русский язык

system("cls"); // чистим консоль перед выводом

int number\_psevdo; // число элементов выборки

double vub\_srednee\_fi; // выборочное среднее выборки

// равномерной fi

double vub\_srednee\_R; // выборочное среднее выборки

// равномерной R

double vub\_srednee; // выборочное среднее выборки

// нормального распределения

double vub\_disp\_fi; // выборочная дисперсия выборки

// равномерной fi

double vub\_disp\_R; // выборочная дисперсия выборки

// равномерной R

double vub\_disp; // выборочная дисперсия выборки

// нормального распределения

int k; // наибольшее количество интервалов

// гистограммы и далее-реальное

int columns = 7; // количество колонок в матрице для

// гистограмм

int begin; // начало динамического массива-вектора

int end; // конец динамического массива -вектора

double\* mass\_raspr = NULL; // указатель на динамический массив

// преобразования. Записываем 0, чтобы

// указатель не был диким

double\* fi = NULL; // указатель на динамический массив

// равномерно распределённой СВ.

// Записываем 0.

double\* R = NULL; // указатель на динамический массив

// равномерно распределённой СВ.

// Записываем 0

double\*\* ppMatrA = NULL; // указатель на матрицу указателей

// преобразования. Записываем 0, чтобы

// указатель не был диким

double\*\* ppMatrfi = NULL; // указатель на матрицу указателей

// равномерно распределённой СВ.

// Записываем 0

double\*\* ppMatrR = NULL; // указатель на матрицу указателей

// равномерно распределённой СВ.

// Записываем 0

fin.open(FNAME\_READ); // связываем объект с файлом.

// Открываем для чтения

if (!fin.is\_open()) // файл не найден

{

cout << "Файла с таким именем нет. Невозможно осуществить

чтение\n";

fin.close(); // закрыть файл для чтения

system("pause");

return 1; // возвращаем значение, соответствующее типу в caller

}// if (!fin.is\_open())

fout.open(FNAME, ios::\_Nocreate); // связываем объект с файлом.

// Открываем для записи

if (!fout.is\_open()) // файл не найден

{

cout << "Файла с таким именем нет. Невозможно осуществить

запись\n";

fout.close(); // закрыть файл для записи

system("pause");

return 2; // возвращаем значение, соответствующее типу в caller

}// if(!fout.is\_open())

cout << "\t\t\t\x1b[35mГАУССОВСКОЕ РАСПРЕДЕЛЕНИЕ С ПОМОЩЬЮ КОСИНУСНОГО ПРЕОБРАЗОВАНИЯ БОКСА-МЮЛЛЕРА\x1b[30m\n\n";

cout << "\x1b[31mЗАМЕЧАНИЕ:\x1b[30m если Вы хотите напечатать подробно сформированную выборку, вводите количество пар не очень большим!\n\n";

{

cout << "Количество пар псевдослучайных чисел, считанное из

файла: ";

fin >> number\_psevdo; // размер массива, введённый пользователем

if (number\_psevdo <=0 ) // если количество элементов не число

{// выход по ошибке

cout << "Количество элементов не является числом или вне

диапазона!!!";

cout << "\n\t\t\t\t\tЗ А В Е Р Ш Е Н И Е П Р О Г Р А М М Ы

\n\n";

system("pause");

return 3; // закончили программу по ошибке

}//if (number\_psevdo <=0)

cout << number\_psevdo << endl << endl;

// ввели количество элементов число - создали массив на нужное

// количество элементов

mass\_raspr = new double[number\_psevdo];

fi = new double[number\_psevdo];

R = new double[number\_psevdo];

}

// заполняем его числами, полученными в результате преобразования

Box\_Muller\_preobr(mass\_raspr, fi, R, number\_psevdo);

char vubor; // выбор, хочет ли пользователь печатать сформированный

// массив на экран

cout << "Хотите ли Вы печатать сформированный результат

преобразования и массивы равномерных процессов?"<<endl<<"Введите Y

или N латиницей : ";

cin >> vubor; // ввод выбора, хочет ли пользователь печатать

cout << "Вы ввели: " << vubor << endl; // эхо -печать

// если пользователь хочет печатать

if (vubor == 'Y' || vubor == 'y')

{

cout << "\n\t\t\t\t \x1b[33mС Ф О Р М И Р О В А Н Н Ы Й М А С С И

В З Н А Ч Е Н И Й fi:\x1b[30m \n";

fout << "\t\t\t\t С Ф О Р М И Р О В А Н Н Ы Й М А С С И В З Н А Ч

Е Н И Й fi: \n";

// печать сформированного массива преобразования

pechat\_tabl(fi, number\_psevdo);

cout << "\n\t\t\t\t \x1b[33mС Ф О Р М И Р О В А Н Н Ы Й М А С С И

В З Н А Ч Е Н И Й R:\x1b[30m \n";

fout << "\t\t\t\t С Ф О Р М И Р О В А Н Н Ы Й М А С С И В З Н А Ч

Е Н И Й R: \n";

// печать сформированного массива равномерно распределённой СВ

pechat\_tabl(R, number\_psevdo);

cout << "\n\t\t\t\t \x1b[33mС Ф О Р М И Р О В А Н Н Ы Й М А С С И

В З Н А Ч Е Н И Й:\x1b[30m \n";

fout << "\t\t\t\t С Ф О Р М И Р О В А Н Н Ы Й М А С С И В З Н А Ч

Е Н И Й: \n";

// печать сформированного массива равномерно распределённой СВ

pechat\_tabl(mass\_raspr, number\_psevdo);

}

else

{

// выводим эту информацию на экран и в файл

cout << "\x1b[34mВы не захотели подробно печатать

сформированные выборки\x1b[30m \n\n";

fout << "Вы не захотели подробно печатать сформированные

выборки\n\n";

}// if (vubor == 'Y' || vubor == 'y')

cout << "\n\n\t\t\t\t\x1b[33mР А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н

Е Г О Д Л Я fi \x1b[30m \n";

fout << "\n\n\t\t\t\t Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д

Л Я fi \n";

cout << "\n\x1b[31mПРАВИЛО:\x1b[30m Чтобы найти выборочную

среднюю по первичным данным, нужно просуммировать все варианты и

разделить полученный результат на объём совокупности.\n\n";

// подсчёт выборочного среднего для массива равномерно распределённой

// СВ

vub\_srednee\_fi = Srednee\_znach(fi, number\_psevdo);

cout << "\n\n\n\t\t\t\t \x1b[33mР А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е

Д Н Е Г О Д Л Я R \x1b[30m \n";

fout << "\n\n\t\t\t\t Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О

Д Л Я R \n";

// подсчёт выборочного среднего для массива равномерно распределённой

// СВ

vub\_srednee\_R = Srednee\_znach(R, number\_psevdo);

cout << "\n\n\n\t\t\t \x1b[33mР А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д

Н Е Г О Д Л Я П Р Е О Б Р А З О В А Н И Я \x1b[30m \n";

fout << "\n\n\t\t\t Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д

Л Я П Р Е О Б Р А З О В А Н И Я \n";

// подсчёт выборочного среднего для массива преобразования

vub\_srednee = Srednee\_znach(mass\_raspr, number\_psevdo);

cout << "\n\n\t\t\t\t\x1b[33mР А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е / Р С И И Д Л Я fi\x1b[30m \n";

fout << "\n\n\t\t\t Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И

Д Л Я fi \n";

cout << "\n\x1b[31mОПРЕДЕЛЕНИЕ:\x1b[30m Выборочная дисперсия –

среднее арифметическое квадратов отклонений всех вариант выборки от

её средней.\n";

cout << "\x1b[31mПРАВИЛО:\x1b[30m Стандартное отклонение выборки -

корень из выборочной дисперсии.\n\n";

// вызвали функцию, рассчитывающую выборочную дисперсию для

// массива равномерно распределённой СВ

vub\_disp\_fi = vub\_dispers(fi, number\_psevdo, vub\_srednee);

cout << "\n\n\t\t\t\t\x1b[33mР А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е

Р С И И Д Л Я R\x1b[30m \n";

fout << "\n\n\t\t\t Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И

Д Л Я R \n";

// вызвали функцию, рассчитывающую выборочную дисперсию для

// массива равномерно распределённой СВ

vub\_disp\_R = vub\_dispers(R, number\_psevdo, vub\_srednee);

cout << "\n\n\t\t\t\x1b[33mР А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р

С И И Д Л Я П Р Е О Б Р З О В А Н И Я\x1b[30m \n";

fout << "\n\n\t\t\t\t\t Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И

Д Л Я П Р Е О Б Р З О В А Н И Я \n";

// вызвали функцию, рассчитывающую выборочную дисперсию для

// массива преобразования

vub\_disp = vub\_dispers(mass\_raspr, number\_psevdo, vub\_srednee);

begin = 0; // массивы начинаются с 0-го элемента

end = number\_psevdo - 1; // заканчиваются на один элемент меньшим

// индексом, чем длина

// отсортировали для расчёта медианы массива равномерно

// распределённой СВ

Merge\_sort\_vozr(fi, begin, end); // вызвали сортировку слиянием

// отсортировали для расчёта медианы массива равномерно

// распределённой СВ

Merge\_sort\_vozr(R, begin, end); // вызвали сортировку слиянием

//отсортировали для расчёта медианы массива преобразования

Merge\_sort\_vozr(mass\_raspr, begin, end); // вызвали сортировку слиянием

// для расчёта медианы массив должен быть отсортирован

cout << "\n\n\n\t\t \x1b[33mР А С Ч Ё Т М Е Д И А Н Ы В А Р И А Ц И О / Н Н О Г О Р Я Д А П Р Е О Б Р А З О В А Н И Я\x1b[30m \n";

fout << "\n\n\n\t\t Р А С Ч Ё Т М Е Д И А Н Ы В А Р И А Ц И О Н Н О Г

О Р Я Д А П Р Е О Б Р А З О В А Н И Я\n";

// даём ряд определений, необходимых для вычисления медианы

cout << "\n\x1b[31mОПРЕДЕЛЕНИЕ:\x1b[30m Медиана вариационного

ряда - значение, которое делит его на две равные части (по количеству

вариант).\n";

cout << "\x1b[31mПРАВИЛО:\x1b[30m Массив с выборкой должен быть

отсортирован.\n\n";

// рассчитали медиану выборки массива преобразования

mediana(mass\_raspr, number\_psevdo);

// модифицированная формула корня. Считаем максимальное количество

// интервалов в гистограмме

k = 10\*sqrt(number\_psevdo); // расчёт количества интервалов с

// округлением в меньшую сторону из-за

// типа int.

{

int i; // для прохода по строкам

// создаём динамическую матрицу из указателей на строки с учётом

// того, что мы передавали указатель в функцию. Их k штук для

// каждого массива

ppMatrfi = new double\* [k];

ppMatrR = new double\* [k];

ppMatrA = new double\* [k];

// идём по всем строкам матрицы

// выделение памяти под каждый элемент строки конкретной длины

for (i = 0; i < k; i++)

{

// на каждую строку матрицы указателей выделяем динамический

// массив длиной columns

ppMatrfi[i] = new double[columns];

ppMatrR[i] = new double[columns];

ppMatrA[i] = new double[columns];

}// for i

}

// рассчитываем параметра матрицы и гистограмм

// расчёт параметров для равномерно распределённой СВ fi

{

cout << "\n\n\t\t \x1b[33m Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О

В А Р И А Ц И О Н Н О Г О Р Я Д А Д Л Я fi\x1b[30m \n\n";

fout << "\n\n\t\t Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А

Ц И О Н Н О Г О Р Я Д А Д Л Я fi \n\n";

// рассчитали интервалы гистограммы для равномерно

// распределённой СВ

gistogramma\_ravnomerno(ppMatrfi, k, fi, number\_psevdo);

pechat\_Matr(ppMatrfi, k); // вывели сформированную матрицу

// равномерно распределённой СВ

cout << "\n\x1b[33m Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У

Н К Ц И И П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я

fi\x1b[30m\n\n";

fout << "\n\x1b Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц

И И П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я fi\n\n";

// расcчитываем теоретическую плотность распределения

// равномерно распределённой fi

teor\_plotnost\_ravn(vub\_disp\_fi, number\_psevdo, ppMatrfi, k);

cout << "\n\n\t\t\x1b[33m Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О

Г О П Р О Ц Е С С А Д Л Я fi\x1b[30m\n\n";

fout << "\n\n\t\t Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П

Р О Ц Е С С А Д Л Я fi\n\n";

cout << "\x1b[31mПРАВИЛО:\x1b[30m Все значения функции, что

правее конца самого левого интервала в гистограмме - имеют

нулевой значение функции\n";

cout << "распределения далее берем крайние значения каждого

интервала - за значение на оси X, накапливаем относительные

частоты;\nw\_n = w[i] + w\_n - значение на оси Y\n\n";

// расчёт псевдослучайного процесса для fi

pract\_raspr(ppMatrfi, k);

cout << "\n\n\t\x1b[33m Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О

П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я

fi\x1b[30m\n\n";

fout << "\n\n\t Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д

О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я fi";

cout << "\x1b[31mЗАМЕЧАНИЕ:\x1b[30m расчёт происходит по

формуле функции распределения равномерного распределения в тех

же точках, что и\nпрактическое;\n\n";

// расчёт теоретического псевдослучайного процесса для fi

teor\_raspr\_ravnom(ppMatrfi, k);

}

// расчёт параметров для равномерно распределённой СВ R

{

cout << "\n\n\t\t \x1b[33m Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О

В А Р И А Ц И О Н Н О Г О Р Я Д А Д Л Я R\x1b[30m \n\n";

fout << "\n\n\t\t Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А

Ц И О Н Н О Г О Р Я Д А Д Л Я R \n\n";

// рассчитали интервалы гистограммы для равномерно

// распределённой СВ R

gistogramma\_ravnomerno(ppMatrR, k, R, number\_psevdo);

// вывели сформированную матрицу равномерно распределённой СВ

// R

pechat\_Matr(ppMatrR, k);

cout << "\n\x1b[33m Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У

Н К Ц И И П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я

R\x1b[30m\n\n";

fout << "\n\x1b Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц

И И П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я R\n\n";

// расcчитываем теоретическую плотность распределения

// равномерно распределённой R

teor\_plotnost\_ravn(vub\_disp\_R, number\_psevdo, ppMatrR, k);

cout << "\n\n\t\t\x1b[33m Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О

Г О П Р О Ц Е С С А Д Л Я R\x1b[30m\n\n";

fout << "\n\n\t\t Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П

Р О Ц Е С С А Д Л Я R\n\n";

cout << "\x1b[31mПРАВИЛО:\x1b[30m Все значения функции, что

правее конца самого левого интервала в гистограмме - имеют

нулевой значение функции\n";

cout << "распределения алее берем крайние значения каждого

интервала - за значение на оси X, накапливаем относительные

частоты;\nw\_n = w[i] + w\_n - значение на оси Y\n\n";

// расчёт псевдослучайного процесса для R

pract\_raspr(ppMatrR, k);

cout << "\n\n\t\x1b[33m Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О

П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я

R\x1b[30m\n\n";

fout << "\n\n\t Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д

О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я R";

cout << "\x1b[31mЗАМЕЧАНИЕ:\x1b[30m расчёт происходит по

формуле функции распределения равномерного распределения в тех

же точках, что и\nпрактическое;\n\n";

// расчёт теоретического псевдослучайного процесса для R

teor\_raspr\_ravnom(ppMatrR, k);

}

// расчёт параметров для преобразования

{

cout << "\n\n \x1b[33m Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В

А Р И А Ц И О Н Н О Г О Р Я Д А Д Л Я П Р Е О Б Р А З О В А Н

И Я\x1b[30m \n\n";

fout << "\n\n Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А Ц

И О Н Н О Г О Р Я Д А Д Л Я П Р Е О Б Р А З О В А Н И Я \n\n";

// рассчитали интервалы гистограммы для преобразования

gistogramma(ppMatrA, k, mass\_raspr, number\_psevdo, vub\_srednee);

// вывели сформированную матрицу преобразования

pechat\_Matr(ppMatrA, k);

cout << "\n\x1b[33m Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У

Н К Ц И И П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я

\n";

cout << "\t\t\t\t\t\tП Р Е О Б Р А З О В А Н И Я\x1b[30m \n";

fout << "\n\n\t\t Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц

И И П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я \n\n";

fout << "\t\t\t\t\t\tП Р Е О Б Р А З О В А Н И Я\n";

// рассчитываем теоретическую плотность распределения

// преобразования

teor\_plotnost\_norm(vub\_srednee, vub\_disp, number\_psevdo, ppMatrA,

k);

cout << "\n\n\t\x1b[33m Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О

Г О П Р О Ц Е С С А Д Л Я П Р Е О Б Р А З О В А Н И

Я\x1b[30m\n\n";

fout << "\n\n\t Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П Р

О Ц Е С С А Д Л Я П Р Е О Б Р А З О В А Н И Я\n\n";

cout << "\x1b[31mПРАВИЛО:\x1b[30m Все значения функции, что

правее конца самого левого интервала в гистограмме - имеют

нулевой значение функции\n";

cout << "распределения алее берем крайние значения каждого

интервала - за значение на оси X, накапливаем относительные

частоты;\nw\_n = w[i] + w\_n - значение F(X)\n\n";

// расчёт псевдослучайного процесса для преобразования

pract\_raspr(ppMatrA, k);

cout << "\n\n\t\x1b[33m Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О

П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л

Я\n\t\t\t\t\t\t П Р Е О Б Р А З О В А Н И Я\x1b[30m\n\n";

fout << "\n\n\t Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д

О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я\n\t\t\t\t\t\t П Р Е О

Б Р А З О В А Н И Я\n\n";

cout << "\x1b[31mЗАМЕЧАНИЕ:\x1b[30m расчёт происходит по

формуле функции распределения гауссовского распределения в тех

же точках, что и\nпрактическое;\n\n";

// расчёт теоретического псевдослучайного процесса для

// преобразования

teor\_raspr\_norm(vub\_srednee, vub\_disp, ppMatrA, k);

}

cout << "\n\n\t\t \x1b[33m Р А С Ч Ё Т М О Д Ы В А Р И А Ц И О Н Н О

Г О Р Я Д А П Р Е О Б Р А З О В А Н И Я\x1b[30m \n\n";

fout << "\n\n\t\t Р А С Ч Ё Т М О Д Ы В А Р И А Ц И О Н Н О Г О Р Я

Д А П Р Е О Б Р А З О В А Н И Я\n\n";

cout << "\n\x1b[31mПРАВИЛО:\x1b[30m Чтобы найти моду, нужно найти

модальный интервал(с максимальной частотой) и воспользоваться

формулой: \n";

cout << "M = X[0]+(n[M]-n[M-1])/((n[M]-n[M-1]) + (n[M]-n[M+1]))\n\n";

// подсчитали моду результата преобразования

moda(ppMatrA, k, columns);

delete\_Matr(ppMatrfi, k); // отработали - удалили матрицу

delete\_Matr(ppMatrR, k); // отработали - удалили матрицу

delete\_Matr(ppMatrA, k); // отработали - удалили матрицу

udalenie(fi); // отработали - удалили массив

udalenie(R); // отработали - удалили массив

udalenie(mass\_raspr); // отработали - удалили массив

fin.close(); // закрыть файл считывания

system("pause");

return 0; // вернули обещанное значение в caller

}

/\*-----------------------------------------------------------------------------\*/

/\* удаление динамического массива \*/

/\*-------------------------------------------------\*/

void udalenie(double\* mass\_raspr) // указатель на первый элемент . . . . . . . . . . . . // динамического массива-вектора

{

delete[] mass\_raspr; // удалили динамический массив-вектор

return; // возвращаем обещанное значение в caller

}// udalenie()

/\*--------------------------------------------------------------------------\*/

/\* печать сформированного массива в таблице \*/

/\*---------------------------------------------------------\*/

void pechat\_tabl(double\* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

, int number) // количество элементов в массиве-векторе

{

int j; // для прохода по элементам массива

setlocale(0, "C"); // отключаем русский язык

cout << (char)218 << (char)196 << (char)196 << setw(30) << setfill(' ') <<

(char)196 << (char)196 << (char)191; // вывод верхних квадратных скобок

fout << (char)218 << (char)196 << (char)196 << setw(30) << setfill(' ') <<

(char)196 << (char)196 << (char)191;

// идем по всем элементам массива, используя адресную арифметику for (j = 0; j < number; j++)

{// выводим элемент массива, боковые скобки и номер элемента

cout << "\n" << char(179) << setw(17) << setfill(' ') <<

\*(mass\_raspr + j) << setw(17) << setfill(' ') << char(179) <<

setw(6) << setfill(' ') << "n = " << j + 1;

// выводим в файл значение сформированного массива и его номер

fout << "\n" << char(179) << setw(17) << setfill(' ') <<

\*(mass\_raspr + j) << setw(17) << setfill(' ') << char(179) <<

setw(6) << setfill(' ') << "n = " << j + 1;;

}// for j

cout << '\n';

fout << '\n';

cout << (char)192 << (char)196 << (char)196 << setw(30) << setfill(' ')

<< (char)196 << (char)196 << (char)217 << endl; // вывод нижних

// квадратных скобок

fout << (char)192 << (char)196 << (char)196 << setw(30) << setfill(' ')

<< (char)196 << (char)196 << (char)217 << endl;

setlocale(LC\_ALL, "Rus"); // подключаем русский язык

return; // напечатали - вернулись в caller

}//pechat\_tabl()

/\*--------------------------------------------------------------------------\*/

/\* вывод полученной матрицы \*/

/\*--------------------------------------\*/

void pechat\_Matr(double\*\* ppMatr // указатель первого элемента матрицы

// указателей

, int k) // количество строк в матрице

{

setlocale(0, "C"); // отключаем русский язык

// выводим шапку таблицы

// верхняя строка шапки

fout << char(218) << setw(25) << setfill((char)196) << (char)194 << setw(15)

<< setfill((char)196) << (char)194 << setw(13) << setfill((char)196) << (char)194 << setw(13) << setfill((char)196) << (char)194 << setw(13) << setfill((char)196) << (char)194 << setw(13) << setfill((char)196) << (char)191 << endl << (char)179;

cout << char(218) << setw(25) << setfill((char)196) << (char)194 << setw(15) << setfill((char)196) << (char)194 << setw(13) << setfill((char)196) << (char)194 << setw(13) << setfill((char)196) << (char)194 << setw(13) << setfill((char)196) << (char)194 << setw(13) << setfill((char)196) << (char)191 << endl << (char)179;

// заполнение шапки с разделителями

fout << " intervals " << (char)179 << " X[i] " << (char)179 << " n[i] " << (char)179 << " n[i]/h " << (char)179 << " w[i] " << (char)179 << " w[i]/h ";

cout << " intervals " << (char)179 << " X[i] " << (char)179 << " n[i] " << (char)179 << " n[i]/h " << (char)179 << " w[i] " << (char)179 << " w[i]/h ";

// нижняя строка шапки

fout << (char)179 << endl << (char)195 << setw(25) << setfill((char)196) << (char)197 << setw(15) << setfill((char)196) << (char)197 << setw(13) << setfill((char)196) << (char)197 << setw(13) << setfill((char)196) << (char)197 << setw(13) << setfill((char)196) << (char)197 << setw(13) << setfill((char)196) << (char)180 << endl;

cout << (char)179 << endl << (char)195 << setw(25) << setfill((char)196) << (char)197 << setw(15) << setfill((char)196) << (char)197 << setw(13) << setfill((char)196) << (char)197 << setw(13) << setfill((char)196) << (char)197 << setw(13) << setfill((char)196) << (char)197 << setw(13) << setfill((char)196) << (char)180 << endl;

// выводим элементы матрицы соответственно их колонкам на экран и в файл

for (int i = 0; i < k; i++)

{

fout << (char)179 << setw(11) << setfill(' ') << ppMatr[i][0] << setw(1) << setfill(' ') << char(179) << setw(10) << setfill(' ') << ppMatr[i][1] << setw(3) << setfill(' ') << (char)179 << setw(12) << setfill(' ') << ppMatr[i][2] << setw(3) << setfill(' ') << (char)179 << setw(7) << setfill(' ') << ppMatr[i][3] << setw(6) << setfill(' ') << (char)179 << setw(10) << setfill(' ') << ppMatr[i][4] << setw(3) << setfill(' ') << (char)179 << setw(10) << setfill(' ') << ppMatr[i][5] << setw(3) << setfill(' ') << (char)179 << setw(12) << setfill(' ') << ppMatr[i][6] << setw(1) << setfill(' ');

fout << (char)179 << endl << (char)195 << setw(25) << setfill((char)196) << (char)197 << setw(15) << setfill((char)196) << (char)197 << setw(13) << setfill((char)196) << (char)197 << setw(13) << setfill((char)196) << (char)197 << setw(13) << setfill((char)196) << (char)197 << setw(13) << setfill((char)196) << (char)180 << endl;

cout << (char)179<< setw(11) << setfill(' ') << ppMatr[i][0] << setw(1) << setfill(' ') << char(179) << setw(10) << setfill(' ') << ppMatr[i][1] << setw(3) << setfill(' ') << (char)179 << setw(12) << setfill(' ') << ppMatr[i][2] << setw(3) << setfill(' ') << (char)179 << setw(7) << setfill(' ') << ppMatr[i][3] << setw(6) << setfill(' ') << (char)179 << setw(10) << setfill(' ') << ppMatr[i][4] << setw(3) << setfill(' ') << (char)179 << setw(10) << setfill(' ') << ppMatr[i][5] << setw(3) << setfill(' ') << (char)179 << setw(12) << setfill(' ') << ppMatr[i][6] << setw(1) << setfill(' ');

cout << (char)179 << endl << (char)195 << setw(25) << setfill((char)196) << (char)197 << setw(15) << setfill((char)196) << (char)197 << setw(13) << setfill((char)196) << (char)197 << setw(13) << setfill((char)196) << (char)197 << setw(13) << setfill((char)196) << (char)197 << setw(13) << setfill((char)196) << (char)180 << endl;

}// for i

setlocale(LC\_ALL, "Rus"); // подключаем русский язык

return; // напечатали - вернулись в caller

}// pechat\_Matr()

/\*--------------------------------------------------------------------------\*/

/\* освободить память матрицы \*/

/\*--------------------------------------\*/

void delete\_Matr(double\*\* ppMatr // указатель на первый элемент матрицы

// указателей

, int k) // количество строк в матрице

{

int i; // счётчик

//освобождение памяти в обратном порядке

for (i = 0; i < k; i++)

{

delete[] ppMatr[i]; // удаляем строки (массивы)

}// for i

delete[] ppMatr; // удаляем массив указателей на строки

return; // удалили - вернулись в caller

}// delete\_Matr()

/\*--------------------------------------------------------------------------\*/

/\* расчёт нормального распределения преобразованием \*/

/\*----------------------------------------------------------------------\*/

void Box\_Muller\_preobr(double\* mass\_raspr // указатель на первый элемент

// массива преобразования

, double\* A // указатель на первый элемент

// динамического массива

// равномерно распределённой СВ

, double\* B // указатель на первый элемент

// динамического массива

// равномерно распределённой СВ

, int number) // количество элементов, которое

// необходимо сформировать

{

int i; // счётчик для заполнения

double fi; // независимая случайная величина, равномерно

// распределённая на интервале (0;1]

double R; // независимая случайная величина, равномерно

// распределённая на интервале (0;1]

double norm; // число, полученное в результате косинусного

// преобразования

for (i = 0; i < number; i++)

{

fi = Ravnomernoe\_gen(); // сгенерировали через функцию первое

// псевдослучайное число

R = Ravnomernoe\_gen(); // сгенерировали через функцию второе

// псевдослучайное число

A[i] = fi; // сохранили значение fi в массив

B[i] = R; // сохранили значение R в массив

// рассчитали по формуле независимую случайную величину,

// распределённую нормально по формуле Бокса-Мюллера

norm = cos(2 \* PI \* R) \* sqrt(-2 \* log(fi));

// положили вычисленную величину в массив

mass\_raspr[i] = norm;

}// for i

return;

}// Box\_Muller\_preobr()

/\*---------------------------------------------------------------------------------------------------\*/

/\* создание равномерно распределённых псевдослучайных чисел \*/

/\*-----------------------------------------------------------------------------------\*/

double Ravnomernoe\_gen(void) // функция ничего не принимает

{

double ravnomerno; // очередное равномерно распределённое

// псевдослучайное число

// RAND\_MAX = 32767

ravnomerno = (rand() + 1) / (double(RAND\_MAX) + 1); // начинается не с 0.

// Самое минимальное число, которое может получиться в выборке: 1/32768

return ravnomerno; // вернули вычисленную величину в caller в

// функцию преобразования

}// Ravnomernoe\_gen()

/\*--------------------------------------------------------------------------\*/

/\* подсчёт выборочного среднего \*/

/\*-----------------------------------------\*/

double Srednee\_znach(double\* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

, int number\_psevdo) // количество элементов в массиве

{

double vub\_srednee = 0; // выборочное среднее

int i; // для прохода в цикле по массиву выборки

// идём по всем элементам массива

for (i = 0; i < number\_psevdo; i++)

{

// вычисляем сумму всех элементов выборки

vub\_srednee = vub\_srednee + mass\_raspr[i];

}// for i

// выводим на экран вспомогательные величины для формулы

cout << "Сумма вариант: " << vub\_srednee<<';' << endl;

cout << "Объём совокупности: " << number\_psevdo<<';' << endl;

// выводим в файл вспомогательные величины для формулы

fout << "Сумма вариант: " << vub\_srednee << ';' << endl;

fout << "Объём совокупности: " << number\_psevdo << ';' << endl;

// делим накопленную сумму выборки на количество элементов выборки

vub\_srednee = vub\_srednee / number\_psevdo;

// вывод на экран выборочного среднего

cout << "Выборочное среднее для данной выборки: " << vub\_srednee << ';'

<< endl;

// вывод в файл выборочного среднего

fout << "Выборочное среднее для данной выборки: " << vub\_srednee << ';'

<< endl;

return vub\_srednee; // подсчитали выборочное среднее - вернули его в caller

}// Srednee\_znach()

/\*--------------------------------------------------------------------------\*/

/\* подсчёт выборочной дисперсии \*/

/\*------------------------------------------\*/

double vub\_dispers(double\* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

, int number\_psevdo // количество элементов в массиве

, double vub\_srednee) // рассчитанное ранее выборочное

// среднее

{

double disp = 0; // переменная, хранящая значение выборочной дисперсии

double standart\_otkl = 0; // переменная, хранящая значение

// среднеквадратичного выборочного

// отклонения

int i; // для цикла

char vubor; // выбор пользователя, хочет ли печатать

cout << "Хотите ли Вы выводить подробно все квадраты отклонений?

Введите Y или N латиницей: ";

cin >> vubor; // ввод пользователем выбора печати

// эхо-печать

cout << "Вы ввели: " << vubor << endl;

// если пользователь хочет печатать квадраты отклонений в таблице

if (vubor == 'Y' || vubor == 'y')

{

setlocale(0, "C"); // отключаем русский язык

// печатаем верхнюю часть шапки таблицы

cout << char(218) << setw(17) << setfill((char)196) << (char)194 <<

setw(23) << setfill((char)196) << (char)191 << endl << (char)179;

// печатаем заполнение шапки таблицы

cout << " X[i] " << (char)179 << " (X[i] - X\_sr)^2 ";

// печатаем нижнюю часть шапки таблицы

cout << (char)179 << endl << (char)195 << setw(17) <<

setfill((char)196) << (char)197 << setw(23) << setfill((char)196) <<

(char)180 << endl;

// выводим элемент в ячейке и его квадрат отклонения в соседней

for (i = 0; i < number\_psevdo; i++)

{

// выводим значение элемента массива и квадрат отклонения

cout << (char)179 << setw(14) << setfill(' ') <<

round(mass\_raspr[i]\*1000)/1000 << setw(3) << setfill(' ') <<

(char)179 << setw(16) << setfill(' ') << round((mass\_raspr[i] –

vub\_srednee)\*1000)/1000 << setw(7) << setfill(' ');

// печатаем строку под очередным выводом в таблице

cout << (char)179 << endl << (char)195 << setw(17) <<

setfill((char)196) << (char)197 << setw(23) << setfill((char)196)

<< (char)180 << endl;

// накапливаем сумму квадратов отклонений

disp = disp + (mass\_raspr[i] - vub\_srednee) \* (mass\_raspr[i] –

vub\_srednee);

}// for i

setlocale(LC\_ALL, "Rus"); // подключаем русский язык

}

// пользователь не хочет печатать таблицу

else

{

// пишем об этом на экране и в файле

cout << "\x1b[34mВы не захотели подробно печатать квадраты

отклонений.\x1b[30m \n\n";

fout << "Вы не захотели подробно печатать квадраты отклонений.

\n\n";

// проделываем то же накопление квадратов отклонений в этом / / / / /

// случае

for (i = 0; i < number\_psevdo; i++)

{

// накапливаем сумму квадратов отклонений

disp = disp + (mass\_raspr[i] - vub\_srednee) \* (mass\_raspr[i] –

vub\_srednee);

}// for i

}// if (vubor == 'Y' || vubor == 'y')

// выводим сумму квадратов отклонений

cout << "Сумма квадратов отклонений: " << disp << ';' << endl;

fout << "Сумма квадратов отклонений: " << disp << ';' << endl;

// делим сумму квадратов отклонений на количество элементов выборки –

// получили выборочную дисперсию

disp = disp / number\_psevdo;

// корень из дисперсии - стандартное отклонение

standart\_otkl = sqrt(disp);

// выводим на экран и в файл полученную выборочную дисперсию и

// стандартное отклонение

cout << "Выборочная дисперсия для данной выборки: " << disp<<';' <<

endl;

cout << "Стандартное отклонение: " << standart\_otkl << ';' << endl;

fout << "Выборочная дисперсия для данной выборки: " << disp << ';' <<

endl;

fout << "Стандартное отклонение: " << standart\_otkl << ';' << endl;

return disp; // подсчитали отклонение и выборочную дисперсию –

// вернули её в caller

}// vub\_dispers()

/\*--------------------------------------------------------------------------\*/

/\* подсчёт медианы выборки \*/

/\*------------------------------------\*/

void mediana(double\* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

, int number\_psevdo) // количество элементов в массиве-векторе

{

// если количество элементов выборки - нечётное

if (number\_psevdo % 2 != 0)

{

// медиана будет целым числом. Элемент, делящий массив на две части

int mediana = 0;

// медиана = n/2

mediana = number\_psevdo / 2;

// выводим на экран промежуточные значения и саму медиану

cout << "\nКоличество чисел выборки " << number\_psevdo << " –

нечётное число. Делим её объём пополам и округляем в

большую сторону\n";

cout << "Индекс медианы выборки: " << mediana<<';' << endl;

cout << "Значение медианы: " << mass\_raspr[mediana]<<';';

// выводим в файл промежуточные значения и саму медиану

fout << "\nКоличество чисел выборки " << number\_psevdo << " –

нечётное число. Делим её объём пополам и округляем в

большую сторону\n";

fout << "Индекс медианы выборки: " << mediana << ';' << endl;

fout << "Значение медианы: " << mass\_raspr[mediana] << ';';

return; // подсчитали медиану - вернулись в caller

}// if (number\_psevdo % 2 != 0)

// количество элементов выборки - четная. Медиана равна среднему от

// двух серединных элементов

double mediana = 0;

// выводим на экран промежуточные значения

cout << "\nКоличество чисел выборки " << number\_psevdo << " - чётное

число. Делим её объём пополам: "<<number\_psevdo/2<<" \n";

cout << "Берём среднее арифметическое " << number\_psevdo / 2 << " и

следующего за ним элемента;\n";

// выводим в файл промежуточные значения

fout << "\nКоличество чисел выборки " << number\_psevdo << " - чётное

число. Делим её объём пополам: " << number\_psevdo / 2 << " \n";

fout << "Берём среднее арифметическое " << number\_psevdo / 2 << " и

следующего за ним элемента;\n";

mediana = (mass\_raspr[(number\_psevdo / 2) - 1]+mass\_raspr[(number\_psevdo

/ 2)])/2;

// выводим на экран саму медиану

cout << "Значение медианы: " << mediana<<';';

// выводим в файл саму медиану

fout << "Значение медианы: " << mediana << ';';

return; // подсчитали медиану - вернулись в caller

}// mediana()

/\*---------------------------------------------------------------------------------\*/

/\* расчёт интервалов и величин гистограммы \*/

/\*---------------------------------------------------------\*/

void gistogramma(double\* ppMatrA[] // указатель первого элемента матрицы

// указателей

, int& k // количество строк матрицы

, double\* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

, int number\_psevdo // количество элементов в массиве

, double vub\_srednee) // рассчитанное выборочное среднее

{

// массив изначально отсортирован

int i; // для цикла

double soxr = 0; // хранение промежуточного значения

double start = mass\_raspr[0]; //наименьший элемент массива - элемент с

//индексом 0. Начало диапазона рассмотрения

// наибольший элемент массива - последний элемент. Конец диапазона

// рассмотрения

double end = mass\_raspr[number\_psevdo - 1];

// количество элементов, попавших в тот или иной интервал

int chislo = 0;

// в идеально построенной выборке для нормального распределения –

// ноль. Считаем центральный интервал, отклоняясь на половину шага от

// нуля в плюс и минус

double minus = round(vub\_srednee) - step / 2;

double kol\_vo\_h; // плотность частот

double w; // относительные частоты

double wi\_h; // плотность относительных частот

k = 0; // для движения по матрице

// двигаемся в последний интервал для отрицательной части

cout << "Минимум: " << start << endl;

cout << "Максимум: " << end << endl;

fout << "Минимум: " << start << endl;

fout << "Максимум: " << end << endl;

// идём от 0 до наименьшего элемента массива. Считаем начало самого

// левого интервала

while (minus > start)

{

// движение с шагом step

minus = minus - step;

}// while

// если модуль начала левого интервала меньше, чем самое большое

// значение элемента массива. Делаем так, чтобы гистограмма начиналась

// и заканчивалась равными по модулю значениями

if (abs(minus) < end)

{

// сдвигаем ещё раз начало

minus = minus - step;

}// if

// запоминаем начало первого интервала гистограммы

soxr = minus;

// ищем количество значений выборки, попавших в интервал. Идём по

// отрицательной части

while (minus <= round(vub\_srednee) - step / 2)

{

// идём по всем элементам массива

for (i = 0; i < number\_psevdo; i++)

{

// элемент попал в интервал

if (minus <= mass\_raspr[i] && mass\_raspr[i] < minus + step)

{

// увеличиваем число элементов в этом интервале

chislo = chislo + 1;

}// if

}// for

// запоминаем начало интервала в 0-ом стролбе матрицы k-ой строки

ppMatrA[k][0] = minus;

// запоминаем конец интервала в 1-ом столбце матрицы k-ой строки

ppMatrA[k][1] = minus + step;

// запоминаем середину интервала в 2-ом столбце матрицы k-ой

// строки

ppMatrA[k][2] = (ppMatrA[k][0] + ppMatrA[k][1]) / 2;

// запоминаем количество элементов в интервале в 3-ом столбце

// матрицы k-ой строки

ppMatrA[k][3] = chislo;

// расчёт плотности частот как частное количества элементов,

// попавших в интервал и длины интервала

kol\_vo\_h = chislo / step;

// запоминаем количество элементов в интервале в 4-ом столбце

// матрицы k-ой строки

ppMatrA[k][4] = kol\_vo\_h;

// сдвигаемся на строку в матрице

k = k + 1;

// сдвигаемся на следующий интервал

minus = minus + step;

// в новом интервале число элементов 0

chislo = 0;

}// while

// далее идём по положительной части массива, начиная со значения, на

// котором остановились

double plus = minus;

// считаем правую границу гистограммы

while (plus < end)

{

// сдвигаемся на значение step

plus = plus + step;

}// while

// если правая граница интервала получилась меньше, чем левая

while (plus < abs(soxr))

{

// увеличиваем границу правого интервала на step

plus = plus + step;

}// while

chislo = 0;

cout << "X[i] - середины интервалов. Вычисляются путем вычитания из

правого конца интервала левого и делением результата на 2.\n";

// ищем количество значений выборки, попавших в интервал. Идём по

// положительной части

while (minus < plus)

{

// идём по всем элементам массива

for (i = 0; i < number\_psevdo; i++)

{

// элемент попал в интервал

if (minus <= mass\_raspr[i] && mass\_raspr[i] < minus+ step)

{

// увеличиваем число элементов в этом интервале

chislo = chislo + 1;

}// if

}// for

// запоминаем начало интервала в 0-ом стролбце матрицы k-ой

// строки

ppMatrA[k][0] = minus;

// запоминаем конец интервала в 1-ом столбце матрицы k-ой строки

ppMatrA[k][1] = minus + step;

// запоминаем середину интервала в 2-ом столбце матрицы k-ой

// строки

ppMatrA[k][2] = (ppMatrA[k][0] + ppMatrA[k][1]) / 2;

// запоминаем количество элементов в интервале в 3-ом столбце

// матрицы k-ой строки

ppMatrA[k][3] = chislo;

// расчёт плотности частот как частное количества элементов,

// попавших в интервал и длины интервала

kol\_vo\_h = chislo / step;

// запоминаем плотности частот элементов в интервале в 4-ом

// столбце матрицы k-ой строки

ppMatrA[k][4] = kol\_vo\_h;

// сдвигаемся на строку в матрице

k = k + 1;

// сдвигаемся на следующий интервал

minus = minus + step;

// в новом интервале число элементов 0

chislo = 0;

}// while

cout << "\x1b[31mОПРЕДЕЛЕНИЕ:\x1b[30m Относительная частота

реализации w[i]- отношение числа случаев, в которых встретился данный

результат,\n";

cout << "к общему числу случаев : n[i] / n.\n";

cout << "\x1b[31mОПРЕДЕЛЕНИЕ:\x1b[30m Плотность относительных

частот - относительная частота, рассчитанная на единицу ширины

интервала: w[i]/h.\n";

cout << "n[i] -количество вариант, попавших в текущий интервал\n";

cout << "\x1b[31mПРАВИЛО:\x1b[30m Если варианта попадает на стык

интервалов, то её следует относить в правый интервал.\n";

cout << "Для всех интервалов, кроме последнего.\n";

cout << "\x1b[31mОПРЕДЕЛЕНИЕ:\x1b[30m Плотность частот - частота, / / рассчитанная на единицу ширины интервала: n[i]/h.\n";

cout << "\x1b[31mОПРЕДЕЛЕНИЕ:\x1b[30m Относительная частота

реализации w[i]- отношение числа случаев, в которых встретился данный

результат,\n";

cout << "к общему числу случаев : n[i] / n.\n";

cout << "\x1b[31mОПРЕДЕЛЕНИЕ:\x1b[30m Плотность относительных

частот - относительная частота, рассчитанная на единицу ширины

интервала: w[i]/h.\n";

// идём по всем строкам матрицы

for (i = 0; i < k; i++)

{

// считаем относительные частоты

w = ppMatrA[i][3] / number\_psevdo;

// помещаем их в шестой столбец матрицы

ppMatrA[i][5] = w;

// считаем плотность относительных частот

wi\_h = w / step;

// помещаем её в седьмой столбец матрицы

ppMatrA[i][6] = wi\_h;

}// for i;

cout << "\n\x1b[34mСАМА МАТРИЦА:\x1b[30m\n\n";

fout << "\nСАМА МАТРИЦА:\n\n";

return; // заполнили матрицу - вернулись в caller

}// gistogramma()

/\*--------------------------------------------------------------------------\*/

/\* расчёт интервалов и величин гистограммы \*/

/\*------------------------------------------\*/

void gistogramma\_ravnomerno(double\* ppMatrA[] // указатель первого

// элемента матрицы указателей

, int& k // количество строк матрицы

, double\* mass\_raspr // указатель на первый элемент

// динамического массива

, int number\_psevdo) // количество элементов в

// массиве-векторе

{

int begin = 0; // начало рассмотрения диапазона гистограммы

int end = 1; // конец рассмотрения диапазона гистограммы

int i; // счётчик движения по массиву

int chislo = 0; // количество элементов, попавших в интервал

double plus = begin; // начало первого интервала

double kol\_vo\_h; // плотность частот

double w = 0; // относительные частоты

double wi\_h = 0; // плотность относительных частот

cout << "\x1b[31mОПРЕДЕЛЕНИЕ:\x1b[30m Относительная частота

реализации w[i]- отношение числа случаев, в которых встретился данный

результат,\n";

cout << "к общему числу случаев : n[i] / n.\n";

cout << "\x1b[31mОПРЕДЕЛЕНИЕ:\x1b[30m Плотность относительных

частот - относительная частота, рассчитанная на единицу ширины

интервала: w[i]/h.\n";

cout << "n[i] -количество вариант, попавших в текущий интервал\n";

cout << "\x1b[31mПРАВИЛО:\x1b[30m Если варианта попадает на стык

интервалов, то её следует относить в правый интервал.\n";

cout << "Для всех интервалов, кроме последнего.\n";

cout << "\x1b[31mОПРЕДЕЛЕНИЕ:\x1b[30m Плотность частот - частота,

рассчитанная на единицу ширины интервала: n[i]/h.\n";

cout << "\x1b[31mОПРЕДЕЛЕНИЕ:\x1b[30m Относительная частота

реализации w[i]- отношение числа случаев, в которых встретился данный

результат,\n";

cout << "к общему числу случаев : n[i] / n.\n";

cout << "\x1b[31mОПРЕДЕЛЕНИЕ:\x1b[30m Плотность относительных

частот - относительная частота, рассчитанная на единицу ширины

интервала: w[i]/h.\n";

k = 0; // для движения по матрице

// пока не дошли до конца интервала

while (plus < end)

{

// движемся по массиву

for (i = 0; i < number\_psevdo; i++)

{

// считаем количество значений, попавших в интервал

if (plus <= mass\_raspr[i] && mass\_raspr[i] < plus +

step\_ravnomerno)

{

chislo = chislo + 1;

}// if

}// for

// запоминаем начало интервала в 0-ом стролбце матрицы k-ой

// строки

ppMatrA[k][0] = plus;

// запоминаем конец интервала в 1-ом столбце матрицы k-ой строки

ppMatrA[k][1] = plus + step\_ravnomerno;

// запоминаем середину интервала в 2-ом столбце матрицы k-ой

// строки

ppMatrA[k][2] = (ppMatrA[k][0] + ppMatrA[k][1]) / 2;

// запоминаем количество элементов в интервале в 3-ом столбце

// матрицы k-ой строки

ppMatrA[k][3] = chislo;

// расчёт плотности частот как частное количества элементов,

// попавших в интервал и длины интервала

kol\_vo\_h = chislo / step\_ravnomerno;

// запоминаем плотности частот элементов в интервале в 4-ом

// столбце матрицы k-ой строки

ppMatrA[k][4] = kol\_vo\_h;

// сдвигаемся на строку в матрице

k = k + 1;

// сдвигаемся на следующий интервал

plus = plus + step\_ravnomerno;

// в новом интервале число элементов 0

chislo = 0;

}// while

for (i = 0; i < k; i++)

{

// считаем относительные частоты

w = ppMatrA[i][3] / number\_psevdo;

// помещаем их в пятый столбец матрицы

ppMatrA[i][5] = w;

// считаем плотность относительных частот

wi\_h = w / step\_ravnomerno;

// помещаем её в шестой столбец матрицы

ppMatrA[i][6] = wi\_h;

}// for i;

cout << "\n\x1b[34mСАМА МАТРИЦА:\x1b[30m\n\n";

fout << "\nСАМА МАТРИЦА:\n\n";

return; // заполнили матрицу - вернулись в caller

}// gistogramma\_ravnomerno()

/\*--------------------------------------------------------------------------\*/

/\* подсчёт моды выборки \*/

/\*--------------------------------\*/

void moda(double\* ppMatrA[] // указатель первого элемента матрицы

// указателей

, int k // количество строк матрицы

, int columns) // количество столбцов матрицы

{

int i; // для прохода по матрице

double max = ppMatrA[0][3]; // максимальная среди частот

// интервалов. Взяли пока первый интервал

int index = 0; // индекс максимальной частоты

int pred\_int; // частота предыдущего интервала от максимального

int sl\_int; // частота следующего интервала от максимального

double h; // длина модального интервала

double moda; // само значение моды

// идём по всем строкам матрицы, начиная со второй

for (i = 1; i < k; i++)

{

// если значение частоты в какой-либо строке больше запомненной

if (ppMatrA[i][3] >= max)

{

max = ppMatrA[i][3]; // ставим это значение за max

index = i; // запоминаем строку

}// if (ppMatrA[i][3] >= max)

}// for i

// печать на экран и в файл вспомогательных значений

cout << "X[0] = " << ppMatrA[index][0] << " - нижняя граница модального

интервала\n";

cout << "n[M] = " << max << " - частота модального интервала\n";

fout << "X[0] = " << ppMatrA[index][0] << " - нижняя граница модального

интервала\n";

fout << "n[M] = " << max << " - частота модального интервала\n";

// по запомненной строке находим длину модального интервала,

// обращаясь к соответствующим столбцам с началом и концом интервала

h = ppMatrA[index][1] - ppMatrA[index][0];

// если модальный интервал - первый

if (index == 0)

{

// у первого интервала нет предыдущего

pred\_int = 0;

}

// иначе запоминаем его предка - интервал не первый

else

{ // запомнили частоту интервала, предыдущего модальному

pred\_int = ppMatrA[index - 1][3];

}// if (index == 0)

// вывели частоту предыдущего интервала на экран и в файл

cout << "n[M-1] = " << pred\_int << " - частота предыдущего интервала\n";

fout << "n[M-1] = " << pred\_int << " - частота предыдущего интервала\n";

// если интервал последний - у него нет следующего за ним

if (index == k - 1)

{

// у последнего интервала нет следующего

sl\_int = 0;

}

// иначе - интервал не последний. У него есть следующий за ним

else

{

// запомнили частоту интервала, следующего за модальным

sl\_int = ppMatrA[index + 1][3];

}// if (index == k - 1)

// вывели частоту следующего интервала на экран и в файл

cout << "n[M+1] = " << sl\_int << " - частота следующего интервала\n";

fout << "n[M+1] = " << sl\_int << " - частота следующего интервала\n";

// считаем значение моды по формуле

moda = ppMatrA[index][0] + (((max - pred\_int) \* h) / ((max - pred\_int) + (max

- sl\_int)));

// печать на экран и в файл значения моды

cout << "МОДА: " << moda<<";\n\n";

fout << "МОДА: " << moda << ";\n\n";

}// moda()

/\*--------------------------------------------------------------------------------------\*/

/\* сортировка слиянием по возрастанию \*/

/\*---------------------------------------------------------\*/

void Merge\_sort\_vozr(double\* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

, int begin // начало динамического массива

, int end) // конец динамического массива

int middle; // отталкиваться будем от середины каждой части,

// разделяя массив на 2 подмассива

// если осталось по одному элементу в каждом подмассиве

// базовое условие

if (begin >= end)

{

return; // вернулись на шаг назад в caller, где в рассматриваемой

// части 2 элемента минимум

}// if(begin >= end)

middle = (begin + end) / 2; // вычислили средний элемент для текущего

// подмассива

Merge\_sort\_vozr(mass\_raspr, begin, middle); // сортировка левой части

Merge\_sort\_vozr(mass\_raspr, middle + 1, end); // сортировка правой части

Merge\_vozr(mass\_raspr, begin, middle, end); // слияние двух частей

}// Merge\_sort\_vozr()

/\*-------------------------------------------------------------------------------------------------\*/

/\* "Властвование" - сливание массивов для Merge\_sort\_vozr \*/

/\*-----------------------------------------------------------------------------\*/

void Merge\_vozr(double\* mass\_raspr // указатель на первый элемент

// динамического массива-вектора

, int begin // начало динамического массива

, int middle // середина динамического массива

, int end) // конец динамического массива

{

// соединение начинается с самых маленьких подмассивов в порядке

// рекурсии

int i; // для цикла

int j = 0; // для цикла

int k; // для цикла

int dlB = middle - begin + 1; // длина левой части подмассива

int dlC = end - middle; // длина правой части подмассива

double\* B = NULL; // объявляем указатель на динамический

массив элементов левой части. Инициализируем его, чтобы не был диким

B = new double[dlB + 1]; // создаём динамический подмассив,

// хранящий элементы левой части исходного

double\* C = NULL; // объявляем указатель на динамический

// массив элементов правой части.

// Инициализируем его, чтобы не был диким

C = new double[dlC + 1]; // создаём динамический подмассив,

// хранящий элементы правой части исходного

// заполняем левую часть

// для B выделено j, т.к не обязательно begin=0

for (i = begin; i <= middle; i++)

{

B[j] = mass\_raspr[i]; // элемент левой части - элемент переданного

// подмассива

j++; // перемещаемся по подмассиву

}// for i

j = 0; // обнулили счётчик для работы с подмассивом C

// заполняем правую часть

// для C выделено j, т.к middle+1!=0

for (i = middle + 1; i <= end; i++)

{

C[j] = mass\_raspr[i]; // элемент правой части - элемент переданного

// подмассива

j++; // перемещаемся по подмассиву

}// for i

B[dlB] = DBL\_MAX; // на последнее место каждого массива ставим

// заведомо больший элемент,

C[dlC] = DBL\_MAX; // чтобы не допускать выхода за пределы массива

i = 0; // ставим все счётчики в начало массивов

j = 0;

for (k = begin; k <= end; k++) // идём по подмассиву

{

// выстраиваем иерархию элементов правой и левой части

if (B[i] <= C[j]) // если элемент правой части больше или равен

// элементу левой части

{

mass\_raspr[k] = B[i]; // помещаем элемент из левой части на

// текущее место подмассива

i = i + 1; // перешли к следующему элементу левой части

// если будет DBL\_MAX, то левая часть выходит из рассмотрения

}

else // если элемент левой части больше элемента правой части

{

mass\_raspr[k] = C[j]; // помещаем элемент из правой части на

// текущее место подмассива

j = j + 1; // перешли к следующему элементу правой части

// если будет INT\_MAX, то правая часть выходит из рассмотрения

}// if (B[i] <= C[j])

}// for k

delete[]B; // подмассивы отработаны - уничтожаем их

delete[]C;

return; // вернули обещанное значение в caller

}// Merge\_vozr()

/\*-----------------------------------------------------------------------------------------------\*/

/\* расчёт теоретической плотности нормального распределения \*/

/\*--------------------------------------------------------------------------------\*/

void teor\_plotnost\_norm(double vub\_srednee // рассчитанное выборочное

// среднее

, double vub\_disp // рассчитанная выборочная

// дисперсия

, int number\_psevdo // количество элементов в выборке

, double\* ppMatrA[] // указатель первого элемента

// матрицы указателей

, int k) // количество строк в матрице

// указателей

{

cout << "\nТеоретические значения функции плотности нормального

распределения по вычисленному выборочному среднему и выборочной

дисперсии\n";

double sigma = sqrt(vub\_disp); // стандартное отклонение

cout << "Стандартное отклонение - корень из дисперсии, sigma = " <<

sigma << endl;

cout<<"\x1b[31mЗАМЕЧАНИЕ:\x1b[30m Построение функции плотности

распределения происходит через формулу функции плотности

нормального\nраспределения\n";

cout << "\x1b[31mПРАВИЛО:\x1b[30m Теоретические частоты

вычисляются в середине каждого интервала гистограммы\n";

// значение функции плотности распределения

double znach1;

// теоретическая частота нормального распределения

double n;

// для движения по строкам матрицы

int i;

setlocale(0, "C"); // отключаем русский язык

// печатаем верхнюю часть шапки таблицы, которая включает значение

// середины интервала, значение функции плотности, теоретическую

// частоту

cout << char(218) << setw(17) << setfill((char)196) << (char)194 << setw(23)

<< setfill((char)196)<<(char)194 << setw(23) << setfill((char)196) <<

(char)191 << endl << (char)179;

// печатаем заполнение шапки таблицы

cout << " X " << (char)179 << " f(X) " << (char)179<<"

n(X) ";

// печатаем нижнюю часть шапки таблицы

cout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) <<

(char)197 << setw(23) << setfill((char)196)<<(char)197 << setw(23) <<

setfill((char)196) << (char)180 << endl;

// печатаем верхнюю часть шапки таблицы в файл

fout << char(218) << setw(17) << setfill((char)196) << (char)194 << setw(23)

<< setfill((char)196) << (char)194 << setw(23) << setfill((char)196) <<

(char)191 << endl << (char)179;

// печатаем заполнение шапки таблицы в файл

fout << " X " << (char)179 << " f(X) " << (char)179 << "

n(X) ";

// печатаем нижнюю часть шапки таблицы в файл

fout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) <<

(char)197 << setw(23) << setfill((char)196) << (char)197 << setw(23) <<

setfill((char)196) << (char)180 << endl;

// считаем для середины каждого интервала

for (i = 0; i < k; i++)

{

// вычисление значения функции плотности в середине каждого

// интервала

znach1 = (1 / (sigma \* sqrt(2 \* PI))) \* exp(-pow(ppMatrA[i][2] –

vub\_srednee, 2) / (2 \* pow(sigma, 2)));

// теоретическое значение частоты

n = ((step \* number\_psevdo) / sigma) \* znach1;

// выводим значение элемента середины интервала, плотности

// распределения в этой точке и теоретической частоты на экран

cout << (char)179 << setw(14) << setfill(' ') << ppMatrA[i][2] << setw(3) << setfill(' ') << (char)179 << setw(16) << setfill(' ') << round((znach1) \* 1000) / 1000 << setw(7) << setfill(' ')<<(char)179 << setw(16) << setfill(' ') << round((n) \* 1000) / 1000 << setw(7) << setfill(' ');

// печатаем строку под очередным выводом в таблице на экране

cout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) << (char)197 << setw(23) << setfill((char)196)<<(char)197 << setw(23) << setfill((char)196) << (char)180 << endl;

// выводим значение элемента середины интервала, плотности

// распределения в этой точке и теоретической частоты в файл

fout << (char)179 << setw(14) << setfill(' ') << ppMatrA[i][2] << setw(3) << setfill(' ') << (char)179 << setw(16) << setfill(' ') << round((znach1) \* 1000) / 1000 << setw(7) << setfill(' ') << (char)179 << setw(16) << setfill(' ') << round((n) \* 1000) / 1000 << setw(7) << setfill(' ');

// печатаем строку под очередным выводом в таблице в файле

fout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) << (char)197 << setw(23) << setfill((char)196) << (char)197 << setw(23) << setfill((char)196) << (char)180 << endl;

}// for i

setlocale(LC\_ALL, "Rus"); // подключаем русский язык

return; // вернули обещанное значение в caller

}// teor\_plotnost\_norm()

/\*--------------------------------------------------------------------------------------------\*/

/\* расчёт теоретической плотности равномерного распределения \*/

/\*----------------------------------------------------------------------------------\*/

void teor\_plotnost\_ravn(double vub\_disp // рассчитанная выборочная

// дисперсия

, int number\_psevdo // количество элементов в выборке

, double\* ppMatrA[] // указатель первого элемента

// матрицы указателей

, int k) // количество строк в матрице

// указателей

{

int a = 0; // начало интервала

int b = 1; // конец интервала

int i; // для движения по матрице

double sigma = sqrt(vub\_disp); // стандартное отклонение

double znach1; // значение функции плотности

double n; // теоретическая частота

cout << "\nТеоретические значения функции плотности равномерного

распределения по формуле плотности нормального распределения\n";

cout << "Стандартное отклонение - корень из дисперсии, sigma = " <<

sigma << endl;

cout << "\x1b[31mПРАВИЛО:\x1b[30m Теоретические частоты

вычисляются в середине каждого интервала гистограммы и

добавляются\nвычисления в 0 и 1. Для равномерного распределения

теоретическая частота вычисляется путём деления размера выборки\nна

количество интервалов\n";

cout << "\x1b[31mПРАВИЛО:\x1b[30m Проверку на соответствие суммы

теоретических частот общему количеству элементов осуществлять

суммой\nзначений, вычисленных только в середине интервалов\n";

setlocale(0, "C"); // отключаем русский язык

// печатаем верхнюю часть шапки таблицы, которая включает значение

// середины интервала, значение функции плотности, теоретическую

// частоту

cout << char(218) << setw(17) << setfill((char)196) << (char)194 << setw(23)

<< setfill((char)196) << (char)194 << setw(23) << setfill((char)196) <<

(char)191 << endl << (char)179;

// печатаем заполнение шапки таблицы

cout << " X " << (char)179 << " f(X) " << (char)179 << "

n(X) ";

// печатаем нижнюю часть шапки таблицы

cout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) <<

(char)197 << setw(23) << setfill((char)196) << (char)197 << setw(23) <<

setfill((char)196) << (char)180 << endl;

// печатаем верхнюю часть шапки таблицы в файл

fout << char(218) << setw(17) << setfill((char)196) << (char)194 << setw(23)

<< setfill((char)196) << (char)194 << setw(23) << setfill((char)196) <<

(char)191 << endl << (char)179;

// печатаем заполнение шапки таблицы в файл

fout << " X " << (char)179 << " f(X) " << (char)179 << "

n(X) ";

// печатаем нижнюю часть шапки таблицы в файл

fout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) <<

(char)197 << setw(23) << setfill((char)196) << (char)197 << setw(23) <<

setfill((char)196) << (char)180 << endl;

// печатаем строку под очередным выводом в таблице

znach1 = (1 / (b - a));

// расчёт теоретической частоты равномерного распределения

n = (number\_psevdo / double(k));

// вычисления в 0 совпадают с любым значение теоретической частоты в

// середине любого промежутка

// выводим значения в 0 на экран и в файл

cout << (char)179 << setw(14) << setfill(' ') << "0" << setw(3) << setfill(' ') <<

(char)179 << setw(16) << setfill(' ') << znach1 << setw(7) << setfill(' ') <<

(char)179 << setw(16) << setfill(' ') << n << setw(7) << setfill(' ');

cout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) <<

(char)197 << setw(23) << setfill((char)196) << (char)197 << setw(23) <<

setfill((char)196) << (char)180 << endl;

fout << (char)179 << setw(14) << setfill(' ') << "0" << setw(3) << setfill(' ') <<

(char)179 << setw(16) << setfill(' ') << znach1 << setw(7) << setfill(' ') <<

(char)179 << setw(16) << setfill(' ') << n << setw(7) << setfill(' ');

fout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) <<

(char)197 << setw(23) << setfill((char)196) << (char)197 << setw(23) <<

setfill((char)196) << (char)180 << endl;

// идём по всем элементам массива

for (i = 0; i < k; i++)

{

// считаем значение плотности по формуле для нормального

// распределения

znach1 = (1 / (b - a));

// теоретическая частота для равномерного распределения

n =( number\_psevdo / double(k));

// значение выборки не попадает в интервал

if (ppMatrA[i][0]<a && ppMatrA[i][0] > b)

{

// значение функции плотности и теоретическая частота равны 0

cout << (char)179 << setw(14) << setfill(' ') << ppMatrA[i][0] << setw(3) << setfill(' ') << (char)179 << setw(16) << setfill(' ') << '0' << setw(7) << setfill(' ') << (char)179 << setw(16) << setfill(' ') << '0' << setw(7) << setfill(' ');

cout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) << (char)197 << setw(23) << setfill((char)196)<< (char)197 << setw(23) << setfill((char)196) << (char)180 << endl;

fout << (char)179 << setw(14) << setfill(' ') << ppMatrA[i][0] << setw(3) << setfill(' ') << (char)179 << setw(16) << setfill(' ') << '0' << setw(7) << setfill(' ') << (char)179 << setw(16) << setfill(' ') << '0' << setw(7) << setfill(' ');

fout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) << (char)197 << setw(23) << setfill((char)196) << (char)197 << setw(23) << setfill((char)196) << (char)180 << endl;

}

// значение выборки попадает в интервал

else if (a <= ppMatrA[i][0] && ppMatrA[i][0] <= b)

{// печатаем ранее вычисленное значение функции и теоретическую

// частоту

cout << (char)179 << setw(14) << setfill(' ') << ppMatrA[i][2] << setw(3) << setfill(' ') << (char)179 << setw(16) << setfill(' ') << round((znach1) \* 1000) / 1000 << setw(7) << setfill(' ') << (char)179 << setw(16) << setfill(' ') << round((n) \* 1000) / 1000 << setw(7) << setfill(' ');

cout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) << (char)197 << setw(23) << setfill((char)196) << (char)197 << setw(23) << setfill((char)196) << (char)180 << endl;

fout << (char)179 << setw(14) << setfill(' ') << ppMatrA[i][2] << setw(3) << setfill(' ') << (char)179 << setw(16) << setfill(' ') << round((znach1) \* 1000) / 1000 << setw(7) << setfill(' ') << (char)179 << setw(16) << setfill(' ') << round((n) \* 1000) / 1000 << setw(7) << setfill(' ');

fout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) << (char)197 << setw(23) << setfill((char)196) << (char)197 << setw(23) << setfill((char)196) << (char)180 << endl;

}// if

}// for i

// вычисление функции плотности в 1. Значение совпадает со значениями,

// вычисленными ранее

cout << (char)179 << setw(14) << setfill(' ') << "1" << setw(3) << setfill(' ') << (char)179 << setw(16) << setfill(' ') << znach1 << setw(7) << setfill(' ') << (char)179 << setw(16) << setfill(' ') << n << setw(7) << setfill(' ');

cout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) << (char)197 << setw(23) << setfill((char)196) << (char)197 << setw(23) << setfill((char)196) << (char)180 << endl;

fout << (char)179 << setw(14) << setfill(' ') << "1" << setw(3) << setfill(' ') << (char)179 << setw(16) << setfill(' ') << znach1 << setw(7) << setfill(' ') << (char)179 << setw(16) << setfill(' ') << n << setw(7) << setfill(' ');

fout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) << (char)197 << setw(23) << setfill((char)196) << (char)197 << setw(23) << setfill((char)196) << (char)180 << endl;

setlocale(LC\_ALL, "Rus"); // подключаем русский язык

return; // вернули обещанное значение в caller

}// teor\_plotnost\_ravn()

/\*----------------------------------------------------------------------------\*/

/\* расчет функции псевдослучайных процессов \*/

/\*-----------------------------------------------------------\*/

void pract\_raspr(double\* ppMatrA[] // указатель первого элемента матрицы

// указателей

, int k) // количество строк в матрице указателей

{

cout << "Построение эмперической функции распределения\n";

cout << "\x1b[31mПРАВИЛО:\x1b[30m Значения формируются вида

(x[прав]; w[i]), где w[i] -относительная частота, которая успела

накопиться\nна всех пройденных интервалах\n";

int i ; // для движения по матрице

double x = ppMatrA[0][0] - 1; // начинаем чуть раньше, чем начинается

// 1-ый интервал

double w\_n = 0; // относительные частоты для очередного

// значения функции распределения

setlocale(0, "C"); // отключаем русский язык

// печать верхней строки шапки таблицы

cout << char(218) << setw(17) << setfill((char)196) << (char)194 << setw(23)

<< setfill((char)196) << (char)191 << endl << (char)179;

// заполнение шапки таблицы включает правое значение интервала и

// значение функции распределения

cout << " X " << (char)179 << " F(X) ";

// печатаем нижнюю часть шапки таблицы

cout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) <<

(char)197 << setw(23) << setfill((char)196) << (char)180 << endl;

fout << char(218) << setw(17) << setfill((char)196) << (char)194 << setw(23)

<< setfill((char)196) << (char)191 << endl << (char)179;

// заполнение шапки таблицы включает правое значение интервала и

// значение функции распределения

fout << " X " << (char)179 << " F(X) ";

// печатаем нижнюю часть шапки таблицы в файл

fout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) <<

(char)197 << setw(23) << setfill((char)196) << (char)180 << endl;

// проходим по части, где функция распределения равна 0;

while (x < ppMatrA[0][1])

{

// выводим значение x и функции распределения, которое равно 0

cout << (char)179 << setw(14) << setfill(' ')<< x << setw(3) << setfill(' ')

<< (char)179 << setw(16) << setfill(' ') << '0' << setw(7) << setfill(' ');

// печатаем строку под очередным выводом в таблице

cout << (char)179 << endl << (char)195 << setw(17) <<

setfill((char)196) << (char)197 << setw(23) << setfill((char)196) <<

(char)180 << endl;

fout << (char)179 << setw(14) << setfill(' ') << x << setw(3) << setfill(' ')

<< (char)179 << setw(16) << setfill(' ') << '0' << setw(7) << setfill(' ');

// печатаем строку под очередным выводом в таблице

fout << (char)179 << endl << (char)195 << setw(17) <<

setfill((char)196) << (char)197 << setw(23) << setfill((char)196) <<

(char)180 << endl;

// сдвигаем x

x = x + step;

}// while

// далее идём по всем строкам матрицы

for (i = 0; i < k; i++)

{

// запоминаем конец очередного интервала гистограммы

x = ppMatrA[i][1];

// накапливаем относительные частоты

w\_n = w\_n + ppMatrA[i][5];

// печатаем значения x и w\_n на экран и в файл

cout << (char)179 << setw(14) << setfill(' ') << x << setw(3) << setfill(' ')

<< (char)179 << setw(16) << setfill(' ') << round(w\_n \* 100) / 100 <<

setw(7) << setfill(' ');

// печатаем строку под очередным выводом в таблице

cout << (char)179 << endl << (char)195 << setw(17) <<

setfill((char)196) << (char)197 << setw(23) << setfill((char)196) <<

(char)180 << endl;

fout << (char)179 << setw(14) << setfill(' ') << x << setw(3) << setfill(' ') / << (char)179 << setw(16) << setfill(' ') << round(w\_n \* 100) / 100 <<

setw(7) << setfill(' ');

// печатаем строку под очередным выводом в таблице

fout << (char)179 << endl << (char)195 << setw(17) <<

setfill((char)196) << (char)197 << setw(23) << setfill((char)196) <<

(char)180 << endl;

}

setlocale(LC\_ALL, "Rus"); // подключаем русский язык

return; // вернули обещанное значение в caller

}// pract\_raspr()

/\*------------------------------------------------------------------------------------------\*/

/\* расчет теоретической функции гауссовского процесса \*/

/\*-----------------------------------------------------------------------\*/

void teor\_raspr\_norm(double vub\_srednee // рассчитанное выборочное среднее

, double vub\_disp // рассчитанная выборочная дисперсия

, double\* ppMatrA[] // указатель первого элемента матрицы

// указателей

, int k) // количество строк в матрице

// указателей

{

int i; // для прохода по матрице

double x = ppMatrA[0][0] - 1; // начинаем чуть раньше, чем начинается

// 1-ый интервал

double w\_n = 0; // относительные частоты для очередного

// значения функции распределения

setlocale(0, "C"); // отключаем русский язык

// печать на экран и в файл

cout << char(218) << setw(17) << setfill((char)196) << (char)194 << setw(23)

<< setfill((char)196) << (char)191 << endl << (char)179;

// заполнение шапки таблицы включает правое значение интервала и

// значение функции распределения

cout << " X " << (char)179 << " F(X) ";

// печатаем нижнюю часть шапки таблицы

cout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) <<

(char)197 << setw(23) << setfill((char)196) << (char)180 << endl;

fout << char(218) << setw(17) << setfill((char)196) << (char)194 << setw(23) / / << setfill((char)196) << (char)191 << endl << (char)179;

// заполнение шапки таблицы включает правое значение интервала и

// значение функции распределения

fout << " X " << (char)179 << " F(X) ";

// печатаем нижнюю часть шапки таблицы

fout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) <<

(char)197 << setw(23) << setfill((char)196) << (char)180 << endl;

// проходим по части, где функция распределения равна 0;

while (x < ppMatrA[0][1])

{

// выводим значение x и функции распределения, которое равно 0

cout << (char)179 << setw(14) << setfill(' ') << x << setw(3) << setfill(' ')

<< (char)179 << setw(16) << setfill(' ') << '0' << setw(7) << setfill(' ');

// печатаем строку под очередным выводом в таблице

cout << (char)179 << endl << (char)195 << setw(17) <<

setfill((char)196) << (char)197 << setw(23) << setfill((char)196) <<

(char)180 << endl;

fout << (char)179 << setw(14) << setfill(' ') << x << setw(3) << setfill(' ')

<< (char)179 << setw(16) << setfill(' ') << '0' << setw(7) << setfill(' ');

// печатаем строку под очередным выводом в таблице

fout << (char)179 << endl << (char)195 << setw(17) <<

setfill((char)196) << (char)197 << setw(23) << setfill((char)196) <<

(char)180 << endl;

// сдвигаем x

x = x + step;

}// while

// далее идём по всем строкам матрицы

for (i = 0; i < k; i++)

{

// запоминаем конец очередного интервала гистограммы

x = ppMatrA[i][1];

// вычисляем значение относительной частоты по формуле для

// нормального распределения

w\_n = 0.5\*(1 + erf((x - vub\_srednee)/sqrt(2\*vub\_disp)));

// печатаем значения на экран и в файл

cout << (char)179 << setw(14) << setfill(' ') << x << setw(3) << setfill(' ')

<< (char)179 << setw(16) << setfill(' ') << round(w\_n \* 100) / 100 <<

setw(7) << setfill(' ');

// печатаем строку под очередным выводом в таблице

cout << (char)179 << endl << (char)195 << setw(17) <<

setfill((char)196) << (char)197 << setw(23) << setfill((char)196) <<

(char)180 << endl;

fout << (char)179 << setw(14) << setfill(' ') << x << setw(3) << setfill(' ')

<< (char)179 << setw(16) << setfill(' ') << round(w\_n \* 100) / 100 <<

setw(7) << setfill(' ');

// печатаем строку под очередным выводом в таблице

fout << (char)179 << endl << (char)195 << setw(17) <<

setfill((char)196) << (char)197 << setw(23) << setfill((char)196) <<

(char)180 << endl;

}// for i

setlocale(LC\_ALL, "Rus"); // подключаем русский язык

return; // вернули обещанное значение в caller

}// teor\_raspr\_norm()

/\*------------------------------------------------------------------------------------------\*/

/\* расчет теоретической функции равномерного процесса \*/

/\*------------------------------------------------------------------------\*/

void teor\_raspr\_ravnom(double\* ppMatrA[] // указатель первого элемента

// матрицы указателей

, int k) // количество строк в матрице

// указателей

{

int a = 0; // начало интервала

int b = 1; // конец интервала

int i; // для движения по строкам матрицы

double x = ppMatrA[0][0] - 1; // начинаем чуть раньше, чем начинается

// 1-ый интервал

double w\_n = 0; // относительные частоты для очередного значения

// функции распределения

setlocale(0, "C"); // отключаем русский язык

// печать на экран и в файл

cout << char(218) << setw(17) << setfill((char)196) << (char)194 << setw(23)

<< setfill((char)196) << (char)191 << endl << (char)179;

// заполнение шапки таблицы включает правое значение интервала и

// значение функции распределения

cout << " X " << (char)179 << " F(X) ";

// печатаем нижнюю часть шапки таблицы

cout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) <<

(char)197 << setw(23) << setfill((char)196) << (char)180 << endl;

fout << char(218) << setw(17) << setfill((char)196) << (char)194 << setw(23)

<< setfill((char)196) << (char)191 << endl << (char)179;

// заполнение шапки таблицы включает правое значение интервала и

// значение функции распределения

fout << " X " << (char)179 << " F(X) ";

// печатаем нижнюю часть шапки таблицы

fout << (char)179 << endl << (char)195 << setw(17) << setfill((char)196) <<

(char)197 << setw(23) << setfill((char)196) << (char)180 << endl;

// проходим по части, где функция распределения равна 0;

while (x < ppMatrA[0][1])

{

// выводим значение x и функции распределения, которое равно 0

cout << (char)179 << setw(14) << setfill(' ') << x << setw(3) << setfill(' ')

<< (char)179 << setw(16) << setfill(' ') << '0' << setw(7) << setfill(' ');

// печатаем строку под очередным выводом в таблице

cout << (char)179 << endl << (char)195 << setw(17) <<

setfill((char)196) << (char)197 << setw(23) << setfill((char)196) <<

(char)180 << endl;

fout << (char)179 << setw(14) << setfill(' ') << x << setw(3) << setfill(' ')

<< (char)179 << setw(16) << setfill(' ') << '0' << setw(7) << setfill(' ');

// печатаем строку под очередным выводом в таблице

fout << (char)179 << endl << (char)195 << setw(17) <<

setfill((char)196) << (char)197 << setw(23) << setfill((char)196) <<

(char)180 << endl;

// сдвигаем x

x = x + step;

}// while

// далее идём по всем строкам матрицы

for (i = 0; i < k; i++)

{

// запоминаем конец очередного интервала гистограммы

x = ppMatrA[i][1];

// вычисляем значение относительной частоты по формуле для

// нормального распределения

w\_n = (x - a)/(b-a);

// печатаем значения на экран и в файл

cout << (char)179 << setw(14) << setfill(' ') << x << setw(3) << setfill(' ')

<< (char)179 << setw(16) << setfill(' ') << round(w\_n \* 1000) / 1000 <<

setw(7) << setfill(' ');

// печатаем строку под очередным выводом в таблице

cout << (char)179 << endl << (char)195 << setw(17) <<

setfill((char)196) << (char)197 << setw(23) << setfill((char)196) <<

(char)180 << endl;

fout << (char)179 << setw(14) << setfill(' ') << x << setw(3) << setfill(' ')

<< (char)179 << setw(16) << setfill(' ') << round(w\_n \* 100) / 100 <<

setw(7) << setfill(' ');

// печатаем строку под очередным выводом в таблице

fout << (char)179 << endl << (char)195 << setw(17) <<

setfill((char)196) << (char)197 << setw(23) << setfill((char)196) <<

(char)180 << endl;

}// for i

setlocale(LC\_ALL, "Rus"); // подключаем русский язык

return; // вернули обещанное значение в caller

}// teor\_raspr\_ravnom()

Код в программном варианте:



**Результаты работы программы**

**2.1** Цель: проверить работоспособность программы при вводе не числа в числе элементов динамического массива. Проверка if(number\_psevdo <=0) в main().

Исходные данные:

В файле считывания хранится значение $;

Ожидаемый результат: Введённое количество элементов не является числом или вне диапазона!!!.

Полученный результат:

ГАУССОВСКОЕ РАСПРЕДЕЛЕНИЕ С ПОМОЩЬЮ КОСИНУСНОГО ПРЕОБРАЗОВАНИЯ БОКСА-МЮЛЛЕРА

ЗАМЕЧАНИЕ: если Вы хотите напечатать подробно сформированную выборку, вводите количество пар не очень большим!

Количество пар псевдослучайных чисел, считанное из файла: Количество элементов не является числом или вне диапазона!!!

З А В Е Р Ш Е Н И Е П Р О Г Р А М М Ы

**2.2** Цель: проверить работоспособность программы при подключении несуществующего файла считывания. Проверка if (!fin.is\_open()) в main().

Исходные данные:

Несуществующий файл с названием “Ne\_sushestv.txt”.

Ожидаемый результат: Файла с таким именем нет. Невозможно осуществить чтение.

Полученный результат:

Файла с таким именем нет. Невозможно осуществить чтение.

**2.3** Цель: проверить работоспособность программы при подключении несуществующего файла вывода. Проверка if (!fout.is\_open()) в main().

Исходные данные:

Несуществующий файл с названием “Ne\_zapis.txt”.

Ожидаемый результат: Файла с таким именем нет. Невозможно осуществить запись.

Полученный результат:

Файла с таким именем нет. Невозможно осуществить запись.

**2.4** Цель: проверить работоспособность программы при введении количества пар псевдослучайных чисел равным 101.

Исходные данные:

В файле считывания находится значение 101.

Ожидаемый результат: Полученное распределение будет близким к нормальному. Исходные встроенные псевдослучайные процессы будут близки к равномерным. Результаты работы выведутся в файл и на экран.

Полученный результат:

ГАУССОВСКОЕ РАСПРЕДЕЛЕНИЕ С ПОМОЩЬЮ КОСИНУСНОГО ПРЕОБРАЗОВАНИЯ БОКСА-МЮЛЛЕРА

ЗАМЕЧАНИЕ: если Вы хотите напечатать подробно сформированную выборку, вводите количество пар не очень большим!

Количество пар псевдослучайных чисел, считанное из файла: 101

Хотите ли Вы печатать сформированный результат преобразования и массивы равномерных процессов?

Введите Y или N латиницей : N

Вы ввели: N

Вы не захотели подробно печатать сформированные выборки

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д Л Я fi

ПРАВИЛО: Чтобы найти выборочную среднюю по первичным данным, нужно просуммировать все варианты и разделить полученный результат на объём совокупности.

Сумма вариант: 51.0312;

Объём совокупности: 101;

Выборочное среднее для данной выборки: 0.50526;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д Л Я R

Сумма вариант: 47.9511;

Объём совокупности: 101;

Выборочное среднее для данной выборки: 0.474763;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д Л Я

П Р Е О Б Р А З О В А Н И Я

Сумма вариант: 6.8881;

Объём совокупности: 101;

Выборочное среднее для данной выборки: 0.0681991;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И Д Л Я fi

ОПРЕДЕЛЕНИЕ: Выборочная дисперсия - среднее арифметическое квадратов отклонений всех вариант выборки от её средней.

ПРАВИЛО: Стандартное отклонение выборки - корень из выборочной дисперсии.

Хотите ли Вы выводить подробно все квадраты отклонений? Введите Y или N латиницей: N

Вы ввели: N

Вы не захотели подробно печатать квадраты отклонений.

Сумма квадратов отклонений: 27.3025;

Выборочная дисперсия для данной выборки: 0.270322;

Стандартное отклонение: 0.519925;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И Д Л Я R

Хотите ли Вы выводить подробно все квадраты отклонений? Введите Y или N латиницей: N

Вы ввели: N

Вы не захотели подробно печатать квадраты отклонений.

Сумма квадратов отклонений: 25.3159;

Выборочная дисперсия для данной выборки: 0.250653;

Стандартное отклонение: 0.500652;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И Д Л Я

П Р Е О Б Р З О В А Н И Я

Хотите ли Вы выводить подробно все квадраты отклонений? Введите Y или N латиницей: N

Вы ввели: N

Вы не захотели подробно печатать квадраты отклонений.

Сумма квадратов отклонений: 113.365;

Выборочная дисперсия для данной выборки: 1.12243;

Стандартное отклонение: 1.05945;

Р А С Ч Ё Т М Е Д И А Н Ы В А Р И А Ц И О Н Н О Г О Р Я Д А

П Р Е О Б Р А З О В А Н И Я

ОПРЕДЕЛЕНИЕ: Медиана вариационного ряда - значение, которое делит его на две равные части (по количеству вариант).

ПРАВИЛО: Массив с выборкой должен быть отсортирован.

Количество чисел выборки 101 - нечётное число. Делим её объём пополам и округляем в большую сторону

Индекс медианы выборки: 50;

Значение медианы: -0.0176288;

Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А Ц И О Н Н О Г О

Р Я Д А Д Л Я fi

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

n[i] -количество вариант, попавших в текущий интервал

ПРАВИЛО: Если варианта попадает на стык интервалов, то её следует относить в правый интервал.

Для всех интервалов, кроме последнего.

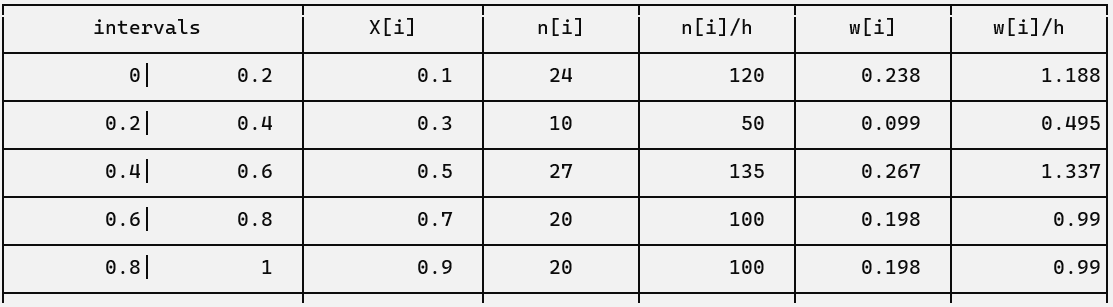
ОПРЕДЕЛЕНИЕ: Плотность частот - частота, рассчитанная на единицу ширины интервала: n[i]/h.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

САМА МАТРИЦА:



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц И И

П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я fi

Теоретические значения функции плотности равномерного распределения по формуле плотности нормального распределения

Стандартное отклонение - корень из дисперсии, sigma = 0.519925

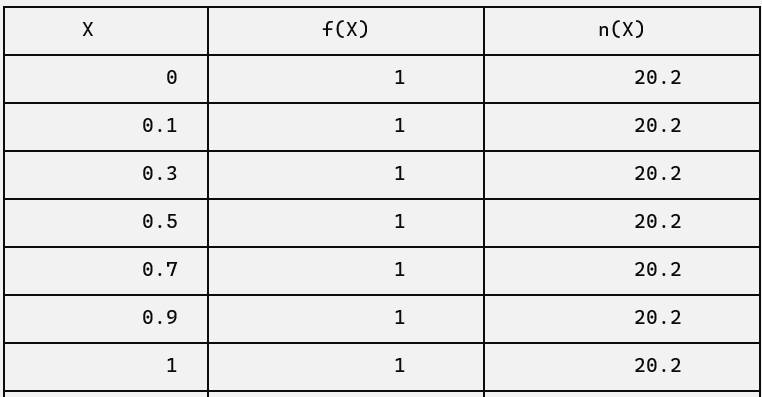
ПРАВИЛО: Теоретические частоты вычисляются в середине каждого интервала гистограммы и добавляются

вычисления в 0 и 1. Для равномерного распределения теоретическая частота вычисляется путём деления размера выборки

на количество интервалов

ПРАВИЛО: Проверку на соответствие суммы теоретических частот общему количеству элементов осуществлять суммой

значений, вычисленных только в середине интервалов



Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А

Д Л Я fi

ПРАВИЛО: Все значения функции, что правее конца самого левого интервала в гистограмме - имеют нулевой значение функции

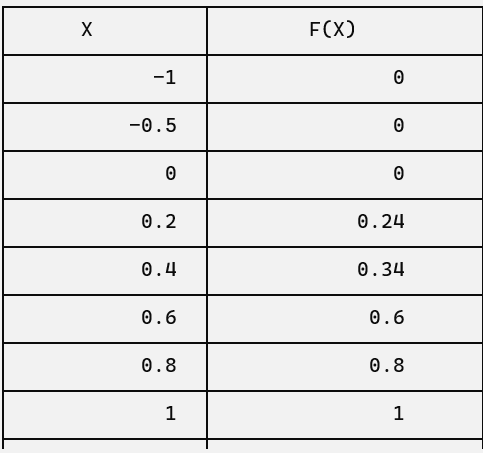
распределения далее берем крайние значения каждого интервала - за значение на оси X, накапливаем относительные частоты;

w\_n = w[i] + w\_n - значение на оси Y

Построение эмперической функции распределения

ПРАВИЛО: Значения формируются вида (x[прав]; w[i]), где w[i] -относительная частота, которая успела накопиться

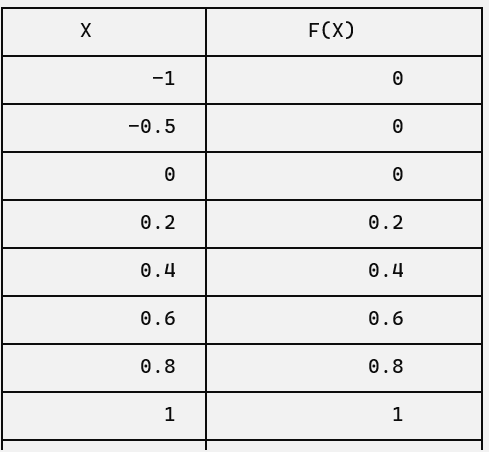
на всех пройденных интервалах



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я fi

ЗАМЕЧАНИЕ: расчёт происходит по формуле функции распределения равномерного распределения в тех же точках, что и

практическое;



Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А Ц И О Н Н О Г О ///////Р Я Д А Д Л Я R

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

n[i] -количество вариант, попавших в текущий интервал

ПРАВИЛО: Если варианта попадает на стык интервалов, то её следует относить в правый интервал.

Для всех интервалов, кроме последнего.

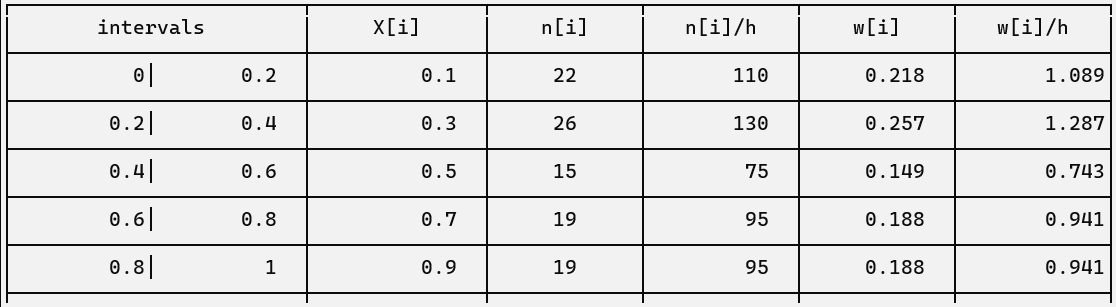
ОПРЕДЕЛЕНИЕ: Плотность частот - частота, рассчитанная на единицу ширины интервала: n[i]/h.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

САМА МАТРИЦА:



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц И И

П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я R

Теоретические значения функции плотности равномерного распределения по формуле плотности нормального распределения

Стандартное отклонение - корень из дисперсии, sigma = 0.500652

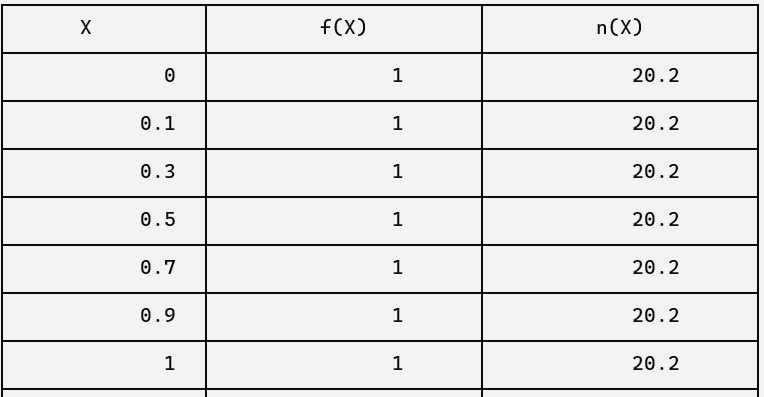
ПРАВИЛО: Теоретические частоты вычисляются в середине каждого интервала гистограммы и добавляются

вычисления в 0 и 1. Для равномерного распределения теоретическая частота вычисляется путём деления размера выборки

на количество интервалов

ПРАВИЛО: Проверку на соответствие суммы теоретических частот общему количеству элементов осуществлять суммой

значений, вычисленных только в середине интервалов



Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я R

ПРАВИЛО: Все значения функции, что правее конца самого левого интервала в гистограмме - имеют нулевой значение функции

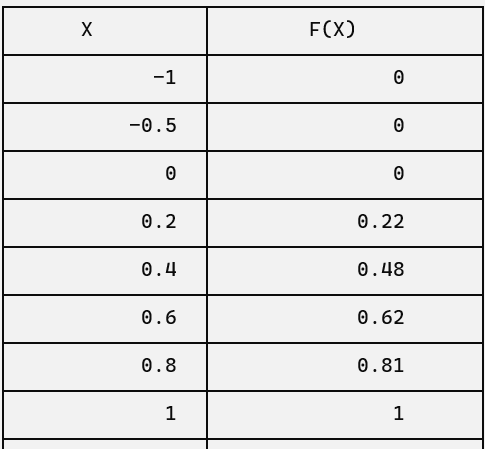
распределения алее берем крайние значения каждого интервала - за значение на оси X, накапливаем относительные частоты;

w\_n = w[i] + w\_n - значение на оси Y

Построение эмперической функции распределения

ПРАВИЛО: Значения формируются вида (x[прав]; w[i]), где w[i] -относительная частота, которая успела накопиться

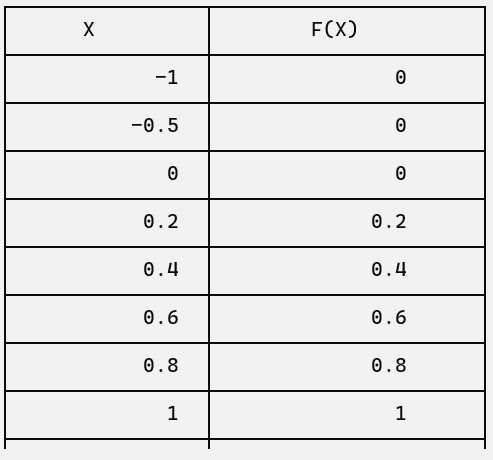
на всех пройденных интервалах



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я R

ЗАМЕЧАНИЕ: расчёт происходит по формуле функции распределения равномерного распределения в тех же точках, что и

практическое;



Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А Ц И О Н Н О Г О

Р Я Д А Д Л Я П Р Е О Б Р А З О В А Н И Я

Минимум: -3.36201

Максимум: 3.31996

X[i] - середины интервалов. Вычисляются путем вычитания из правого конца интервала левого и делением результата на 2.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

n[i] -количество вариант, попавших в текущий интервал

ПРАВИЛО: Если варианта попадает на стык интервалов, то её следует относить в правый интервал.

Для всех интервалов, кроме последнего.

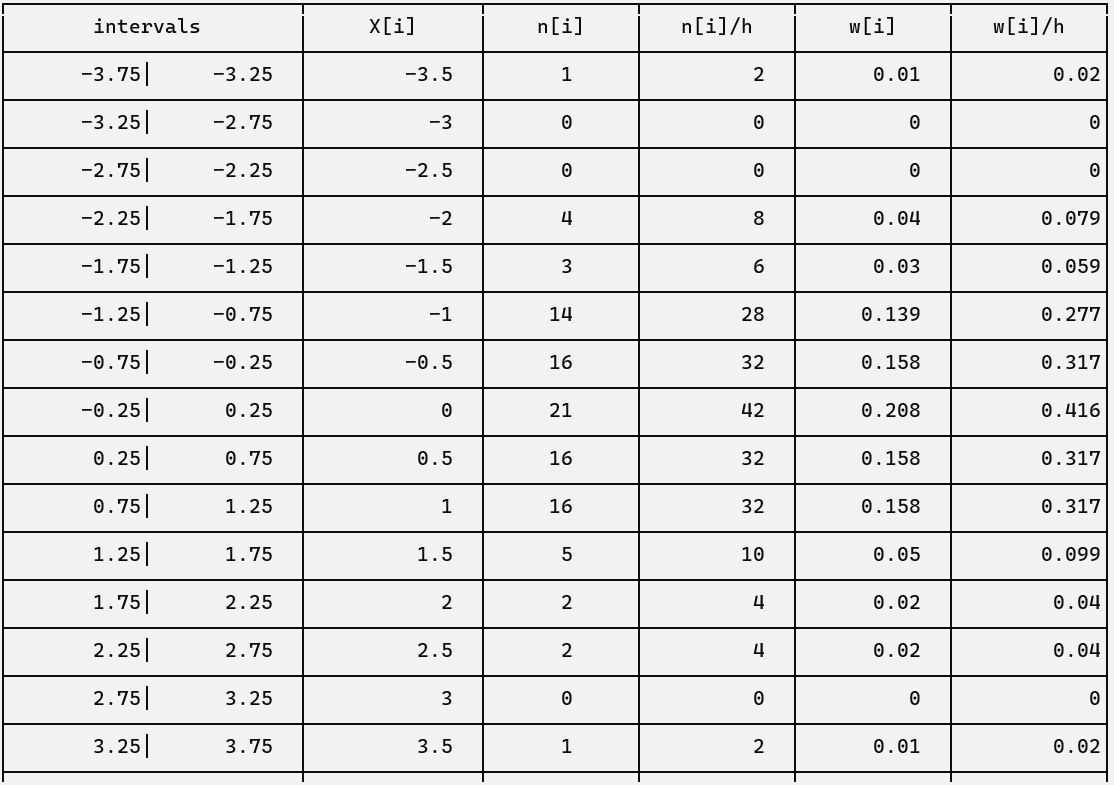
ОПРЕДЕЛЕНИЕ: Плотность частот - частота, рассчитанная на единицу ширины интервала: n[i]/h.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

САМА МАТРИЦА:



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц И И

П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я

П Р Е О Б Р А З О В А Н И Я

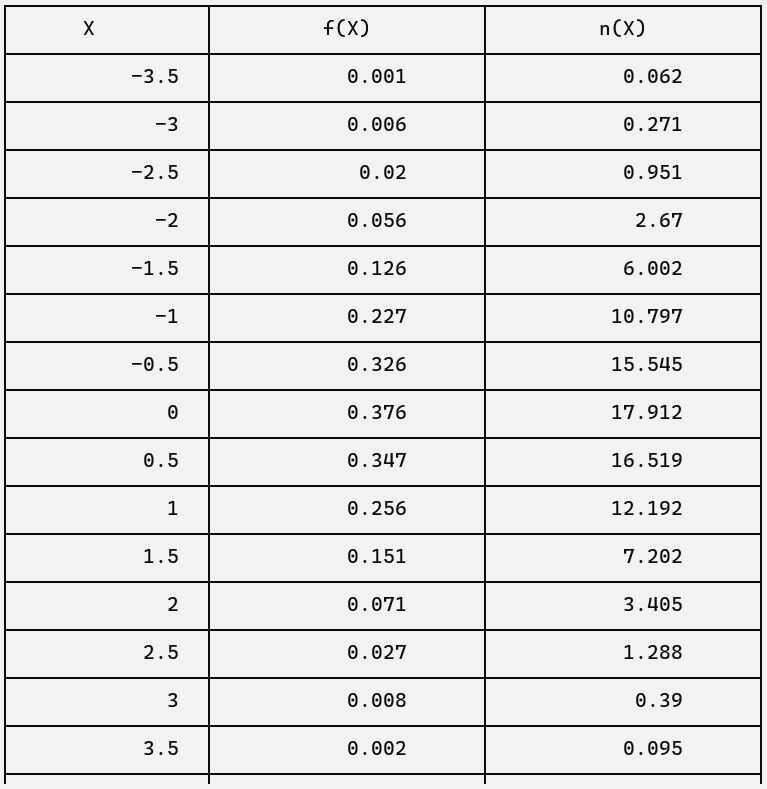
Теоретические значения функции плотности нормального распределения по вычисленному выборочному среднему и выборочной дисперсии

Стандартное отклонение - корень из дисперсии, sigma = 1.05945

ЗАМЕЧАНИЕ: Построение функции плотности распределения происходит через формулу функции плотности нормального

распределения

ПРАВИЛО: Теоретические частоты вычисляются в середине каждого интервала гистограммы



Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я

П Р Е О Б Р А З О В А Н И Я

ПРАВИЛО: Все значения функции, что правее конца самого левого интервала в гистограмме - имеют нулевой значение функции

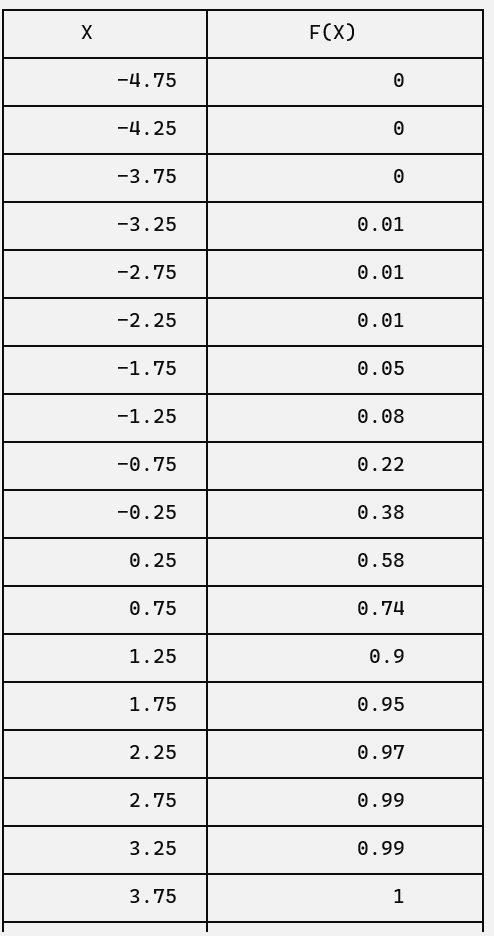
распределения алее берем крайние значения каждого интервала - за значение на оси X, накапливаем относительные частоты;

w\_n = w[i] + w\_n - значение F(X)

Построение эмперической функции распределения

ПРАВИЛО: Значения формируются вида (x[прав]; w[i]), где w[i] -относительная частота, которая успела накопиться

на всех пройденных интервалах

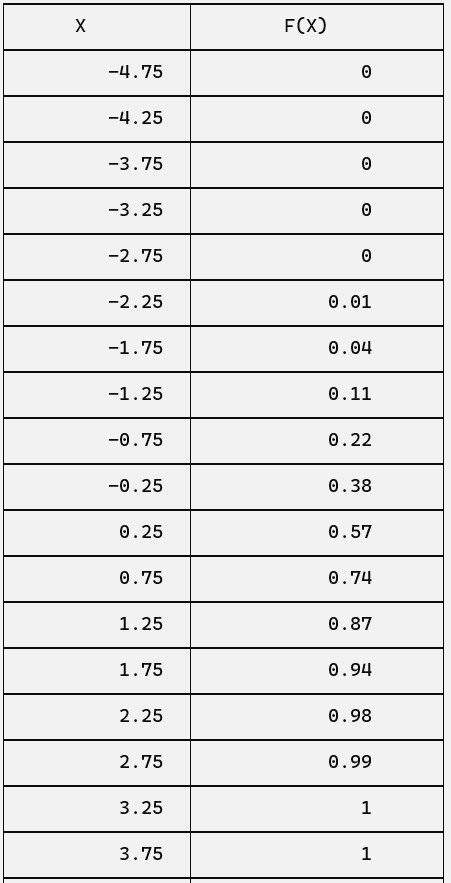


Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я

П Р Е О Б Р А З О В А Н И Я

ЗАМЕЧАНИЕ: расчёт происходит по формуле функции распределения гауссовского распределения в тех же точках, что и

практическое;



Р А С Ч Ё Т М О Д Ы В А Р И А Ц И О Н Н О Г О Р Я Д А //////////////////

/П Р Е О Б Р А З О В А Н И Я

ПРАВИЛО: Чтобы найти моду, нужно найти модальный интервал(с максимальной частотой) и воспользоваться формулой:

M = X[0]+(n[M]-n[M-1])/((n[M]-n[M-1]) + (n[M]-n[M+1]))

X[0] = -0.25 - нижняя граница модального интервала

n[M] = 21 - частота модального интервала

n[M-1] = 16 - частота предыдущего интервала

n[M+1] = 16 - частота следующего интервала

МОДА: 0;

Гистограмма полученного преобразования:

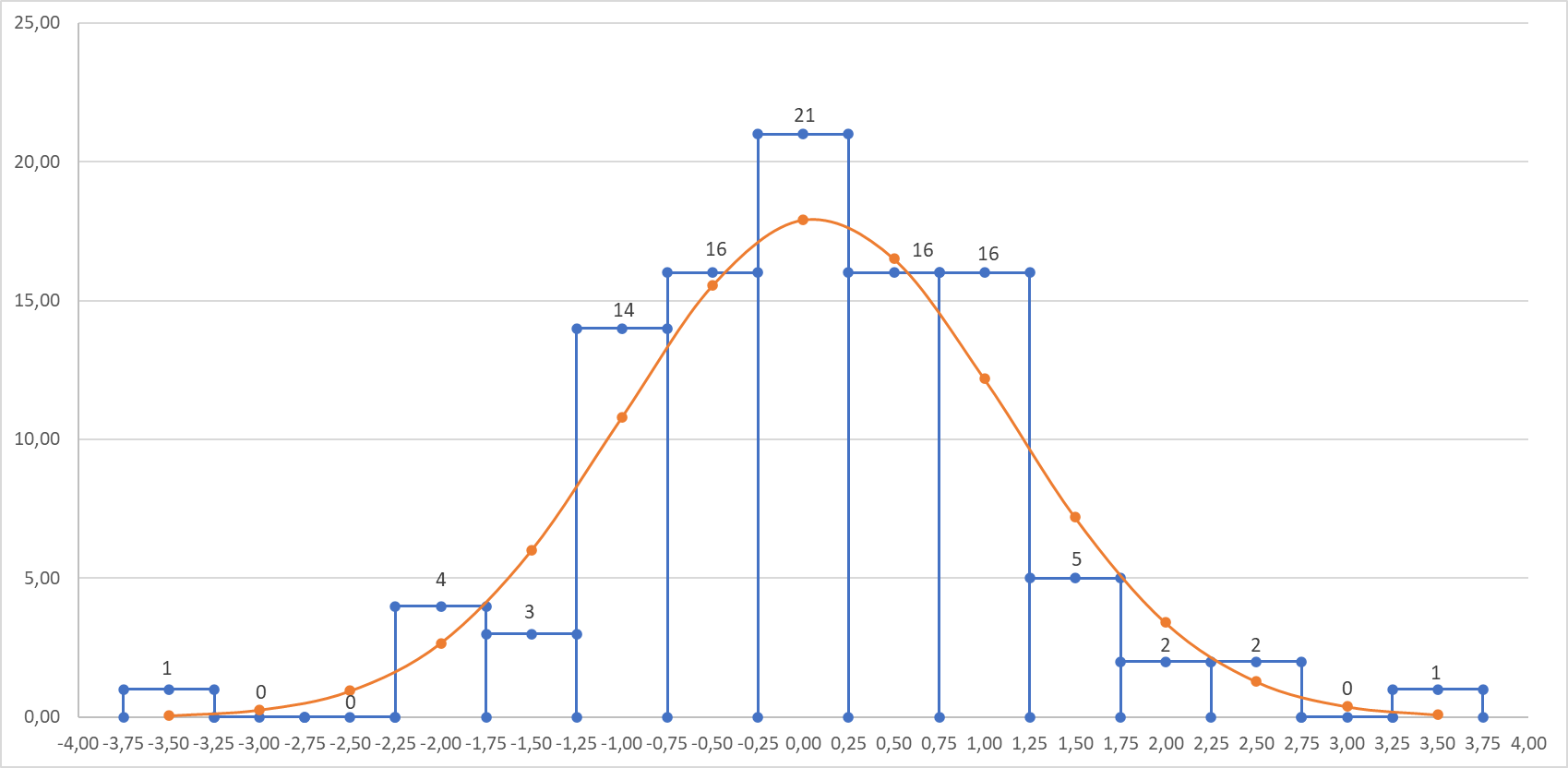
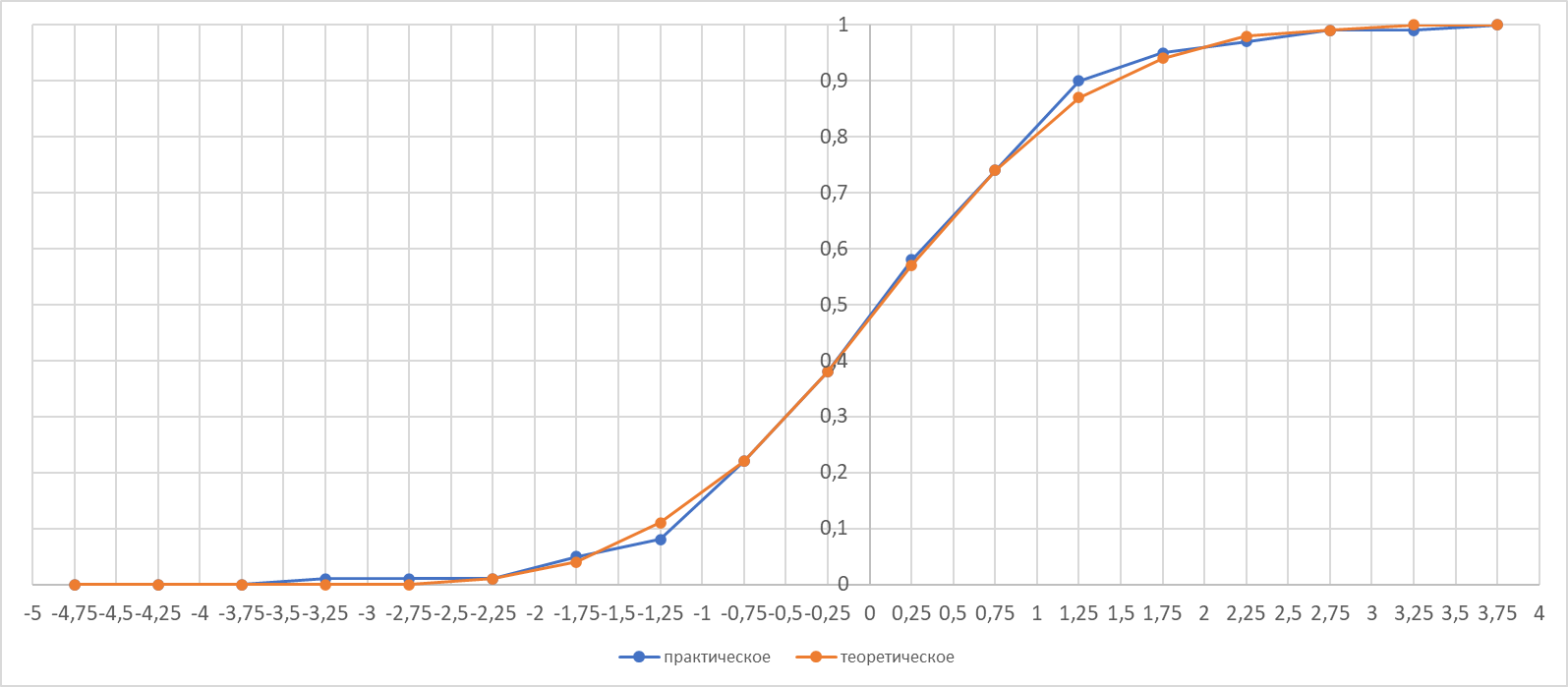


График гауссовского псевдослучайного процесса:



Гистограмма равномерно распределённой СВ fi:

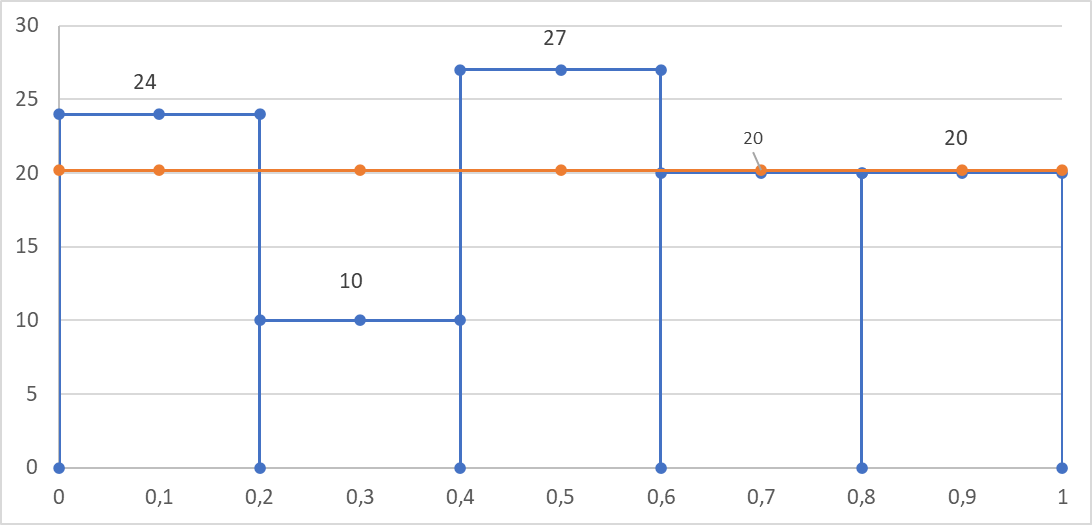
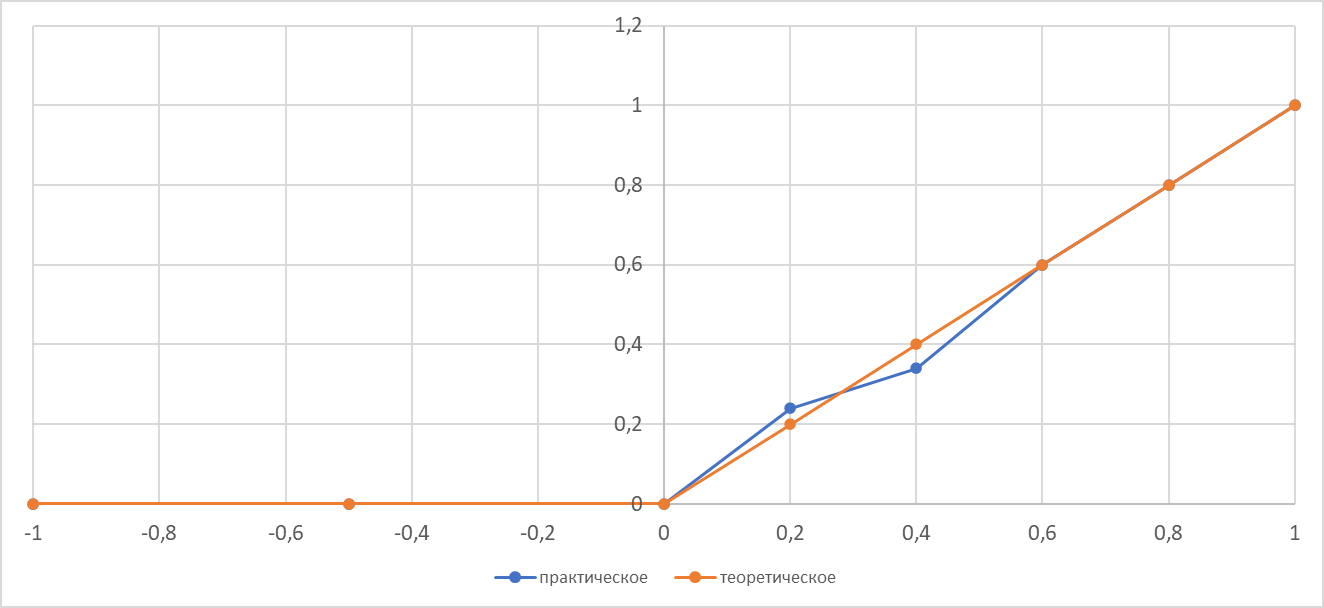


График псевдослучайного равномерного процесса СВ fi:



Гистограмма равномерно распределённой СВ R:

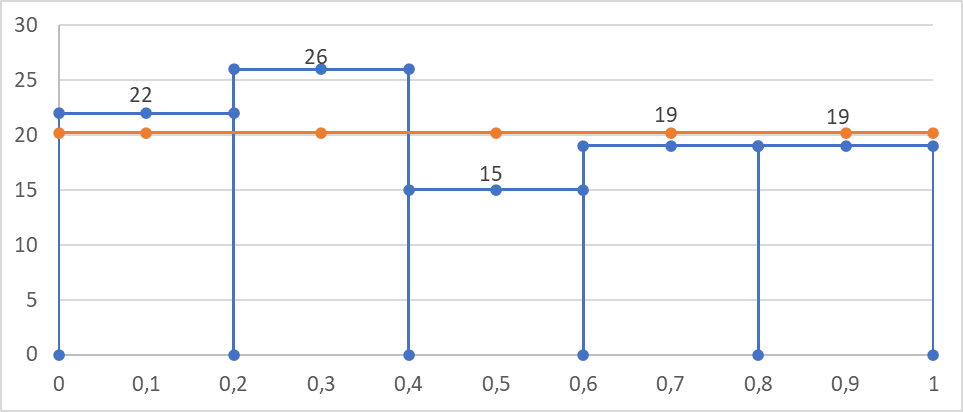
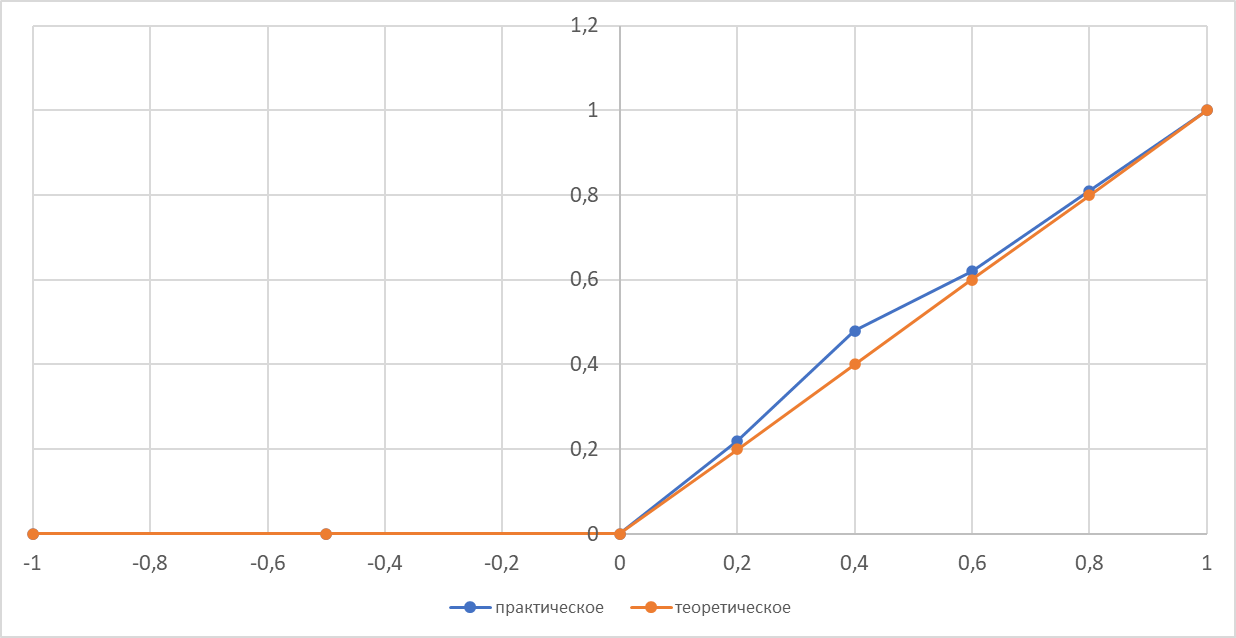


График псевдослучайного равномерного процесса СВ R:



**2.5** Цель: проверить работоспособность программы при введении количества пар псевдослучайных чисел равным 512.

Исходные данные:

В файле считывания находится значение 512.

Ожидаемый результат: Полученное распределение будет близким к нормальному. Исходные встроенные псевдослучайные процессы будут близки к равномерным. Результаты работы выведутся в файл и на экран.

Полученный результат:

ГАУССОВСКОЕ РАСПРЕДЕЛЕНИЕ С ПОМОЩЬЮ КОСИНУСНОГО ПРЕОБРАЗОВАНИЯ БОКСА-МЮЛЛЕРА

ЗАМЕЧАНИЕ: если Вы хотите напечатать подробно сформированную выборку, вводите количество пар не очень большим!

Количество пар псевдослучайных чисел, считанное из файла: 512

Хотите ли Вы печатать сформированный результат преобразования и массивы равномерных процессов?

Введите Y или N латиницей : N

Вы ввели: N

Вы не захотели подробно печатать сформированные выборки

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д Л Я fi

ПРАВИЛО: Чтобы найти выборочную среднюю по первичным данным, нужно просуммировать все варианты и разделить полученный результат на объём совокупности.

Сумма вариант: 252.945;

Объём совокупности: 512;

Выборочное среднее для данной выборки: 0.494034;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д Л Я R

Сумма вариант: 259.69;

Объём совокупности: 512;

Выборочное среднее для данной выборки: 0.507206;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д Л Я

П Р Е О Б Р А З О В А Н И Я

Сумма вариант: -8.71304;

Объём совокупности: 512;

Выборочное среднее для данной выборки: -0.0170177;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И Д Л Я fi

ОПРЕДЕЛЕНИЕ: Выборочная дисперсия - среднее арифметическое квадратов отклонений всех вариант выборки от её средней.

ПРАВИЛО: Стандартное отклонение выборки - корень из выборочной дисперсии.

Хотите ли Вы выводить подробно все квадраты отклонений? Введите Y или N латиницей: N

Вы ввели: N

Вы не захотели подробно печатать квадраты отклонений.

Сумма квадратов отклонений: 176.302;

Выборочная дисперсия для данной выборки: 0.34434;

Стандартное отклонение: 0.586805;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И Д Л Я R

Хотите ли Вы выводить подробно все квадраты отклонений? Введите Y или N латиницей: N

Вы ввели: N

Вы не захотели подробно печатать квадраты отклонений.

Сумма квадратов отклонений: 183.429;

Выборочная дисперсия для данной выборки: 0.358261;

Стандартное отклонение: 0.598549;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И Д Л Я П Р Е О Б Р З О В А Н И Я

Хотите ли Вы выводить подробно все квадраты отклонений? Введите Y или N латиницей: N

Вы ввели: N

Вы не захотели подробно печатать квадраты отклонений.

Сумма квадратов отклонений: 531.082;

Выборочная дисперсия для данной выборки: 1.03727;

Стандартное отклонение: 1.01846;

Р А С Ч Ё Т М Е Д И А Н Ы В А Р И А Ц И О Н Н О Г О Р Я Д А

П Р Е О Б Р А З О В А Н И Я

ОПРЕДЕЛЕНИЕ: Медиана вариационного ряда - значение, которое делит его на две равные части (по количеству вариант).

ПРАВИЛО: Массив с выборкой должен быть отсортирован.

Количество чисел выборки 512 - чётное число. Делим её объём пополам: 256

Берём среднее арифметическое 256 и следующего за ним элемента;

Значение медианы: 0.00823474;

Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А Ц И О Н Н О Г О

Р Я Д А Д Л Я fi

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

n[i] -количество вариант, попавших в текущий интервал

ПРАВИЛО: Если варианта попадает на стык интервалов, то её следует относить в правый интервал.

Для всех интервалов, кроме последнего.

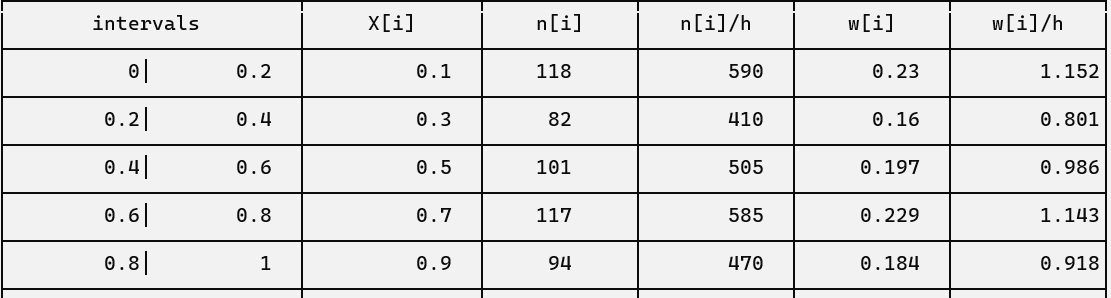
ОПРЕДЕЛЕНИЕ: Плотность частот - частота, рассчитанная на единицу ширины интервала: n[i]/h.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

САМА МАТРИЦА:



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц И И /////////////////////П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я fi

Теоретические значения функции плотности равномерного распределения по формуле плотности нормального распределения

Стандартное отклонение - корень из дисперсии, sigma = 0.586805

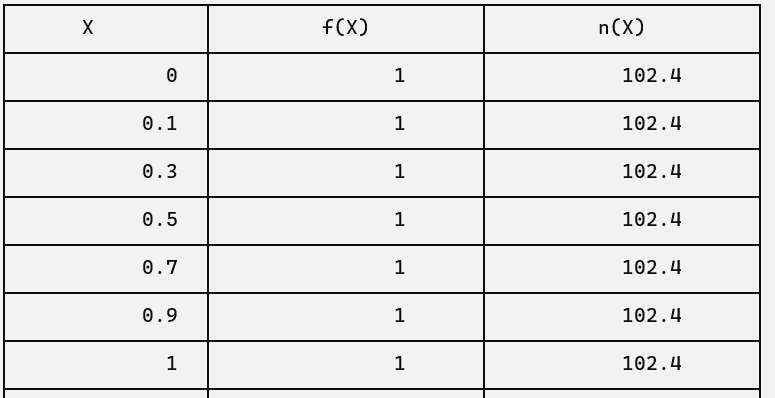
ПРАВИЛО: Теоретические частоты вычисляются в середине каждого интервала гистограммы и добавляются

вычисления в 0 и 1. Для равномерного распределения теоретическая частота вычисляется путём деления размера выборки

на количество интервалов

ПРАВИЛО: Проверку на соответствие суммы теоретических частот общему количеству элементов осуществлять суммой

значений, вычисленных только в середине интервалов



Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я fi

ПРАВИЛО: Все значения функции, что правее конца самого левого интервала в гистограмме - имеют нулевой значение функции

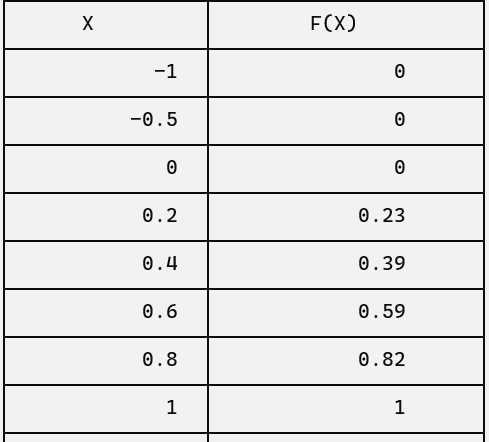
распределения далее берем крайние значения каждого интервала - за значение на оси X, накапливаем относительные частоты;

w\_n = w[i] + w\_n - значение на оси Y

Построение эмперической функции распределения

ПРАВИЛО: Значения формируются вида (x[прав]; w[i]), где w[i] -относительная частота, которая успела накопиться

на всех пройденных интервалах

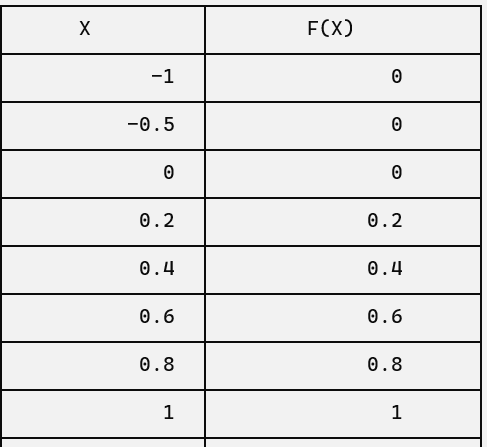


Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О

П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я fi

ЗАМЕЧАНИЕ: расчёт происходит по формуле функции распределения равномерного распределения в тех же точках, что и

практическое;



Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О

В А Р И А Ц И О Н Н О Г О Р Я Д А Д Л Я R

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

n[i] -количество вариант, попавших в текущий интервал

ПРАВИЛО: Если варианта попадает на стык интервалов, то её следует относить в правый интервал.

Для всех интервалов, кроме последнего.

ОПРЕДЕЛЕНИЕ: Плотность частот - частота, рассчитанная на единицу ширины интервала: n[i]/h.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

САМА МАТРИЦА:



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц И И

П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я R

Теоретические значения функции плотности равномерного распределения по формуле плотности нормального распределения

Стандартное отклонение - корень из дисперсии, sigma = 0.598549

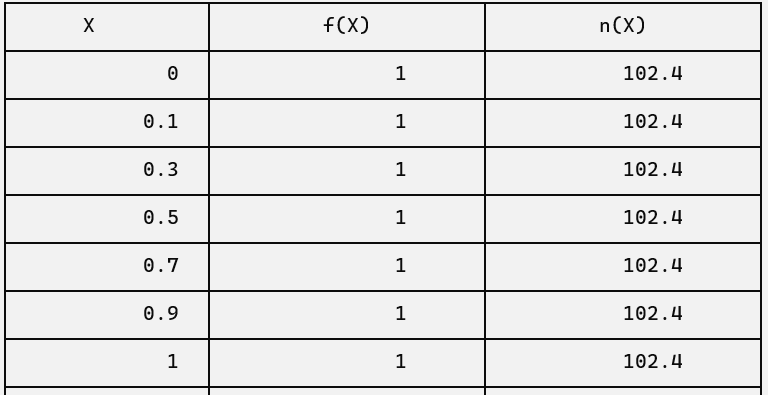
ПРАВИЛО: Теоретические частоты вычисляются в середине каждого интервала гистограммы и добавляются

вычисления в 0 и 1. Для равномерного распределения теоретическая частота вычисляется путём деления размера выборки

на количество интервалов

ПРАВИЛО: Проверку на соответствие суммы теоретических частот общему количеству элементов осуществлять суммой

значений, вычисленных только в середине интервалов



Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я R

ПРАВИЛО: Все значения функции, что правее конца самого левого интервала в гистограмме - имеют нулевой значение функции

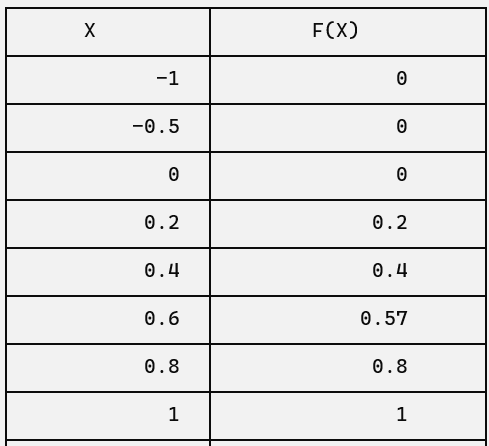
распределения алее берем крайние значения каждого интервала - за значение на оси X, накапливаем относительные частоты;

w\_n = w[i] + w\_n - значение на оси Y

Построение эмперической функции распределения

ПРАВИЛО: Значения формируются вида (x[прав]; w[i]), где w[i] -относительная частота, которая успела накопиться

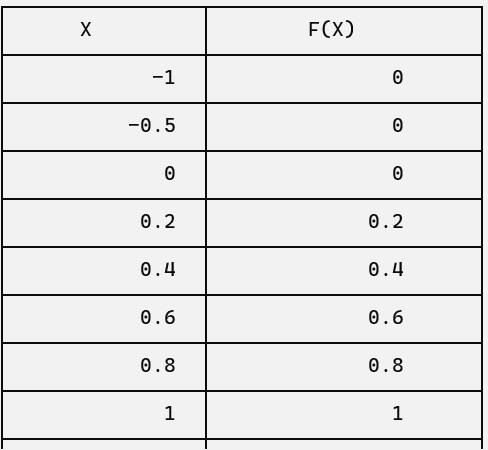
на всех пройденных интервалах



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я R

ЗАМЕЧАНИЕ: расчёт происходит по формуле функции распределения равномерного распределения в тех же точках, что и

практическое;



Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А Ц И О Н Н О Г О

Р Я Д А Д Л Я П Р Е О Б Р А З О В А Н И Я

Минимум: -3.37166

Максимум: 3.31996

X[i] - середины интервалов. Вычисляются путем вычитания из правого конца интервала левого и делением результата на 2.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

n[i] -количество вариант, попавших в текущий интервал

ПРАВИЛО: Если варианта попадает на стык интервалов, то её следует относить в правый интервал.

Для всех интервалов, кроме последнего.

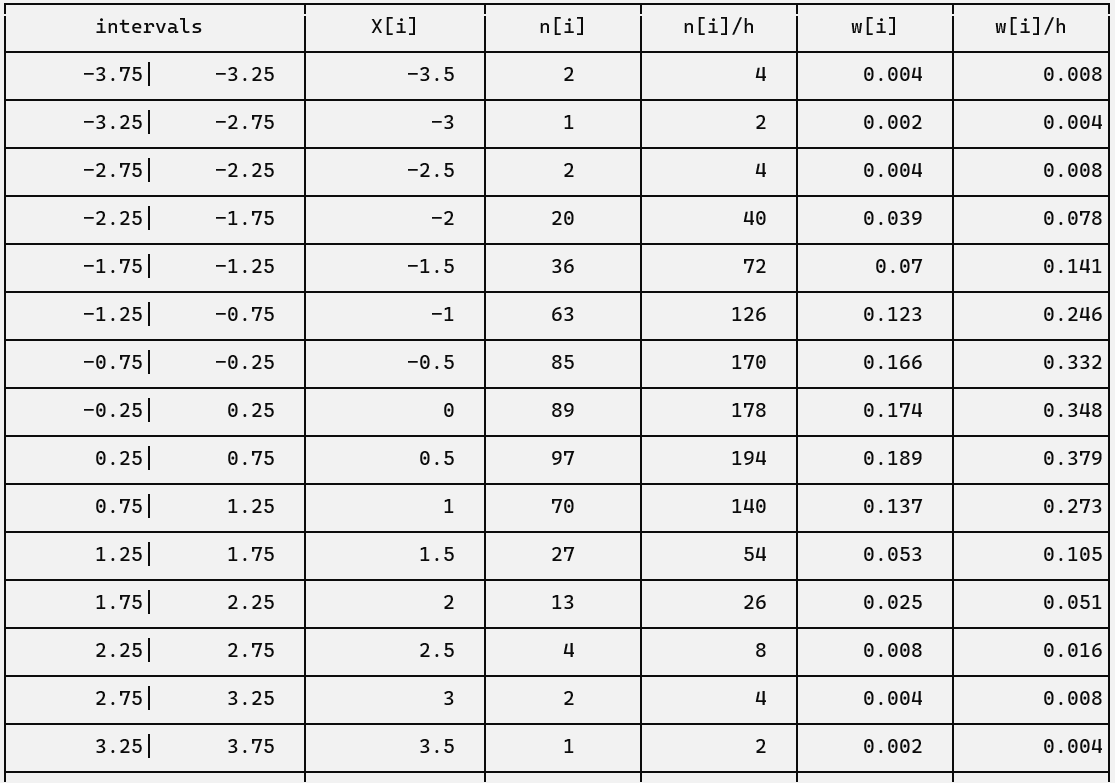
ОПРЕДЕЛЕНИЕ: Плотность частот - частота, рассчитанная на единицу ширины интервала: n[i]/h.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

САМА МАТРИЦА:



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц И И

П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я

П Р Е О Б Р А З О В А Н И Я

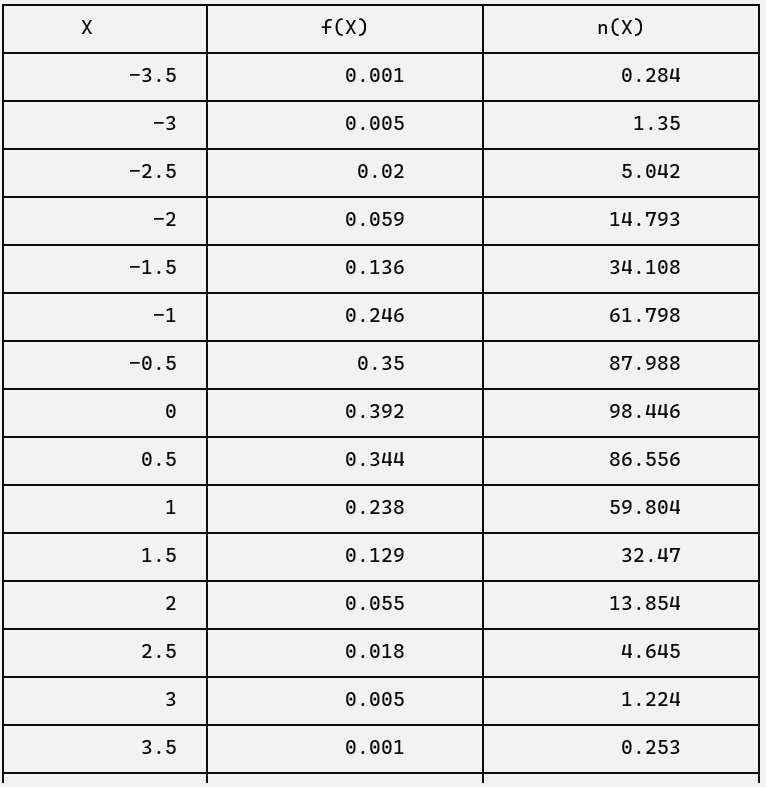
Теоретические значения функции плотности нормального распределения по вычисленному выборочному среднему и выборочной дисперсии

Стандартное отклонение - корень из дисперсии, sigma = 1.01846

ЗАМЕЧАНИЕ: Построение функции плотности распределения происходит через формулу функции плотности нормального

распределения

ПРАВИЛО: Теоретические частоты вычисляются в середине каждого интервала гистограммы



Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я

П Р Е О Б Р А З О В А Н И Я

ПРАВИЛО: Все значения функции, что правее конца самого левого интервала в гистограмме - имеют нулевой значение функции

распределения алее берем крайние значения каждого интервала - за значение на оси X, накапливаем относительные частоты;

w\_n = w[i] + w\_n - значение F(X)

Построение эмперической функции распределения

ПРАВИЛО: Значения формируются вида (x[прав]; w[i]), где w[i] -относительная частота, которая успела накопиться

на всех пройденных интервалах

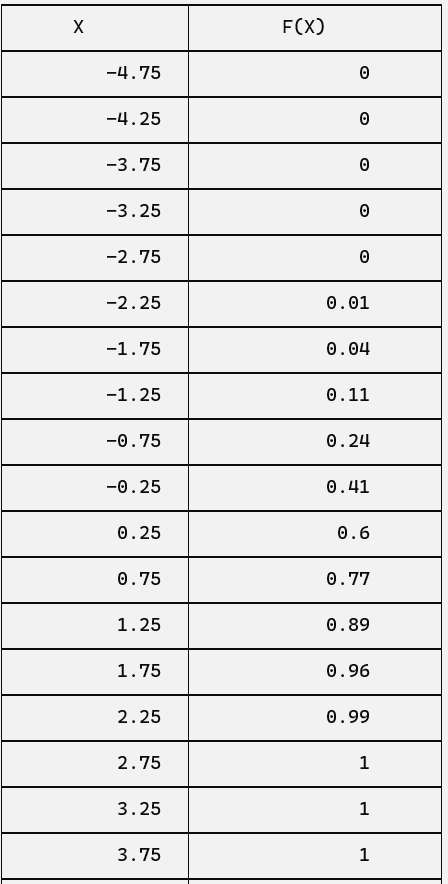


Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я

П Р Е О Б Р А З О В А Н И Я

ЗАМЕЧАНИЕ: расчёт происходит по формуле функции распределения гауссовского распределения в тех же точках, что и

практическое;



Р А С Ч Ё Т М О Д Ы В А Р И А Ц И О Н Н О Г О Р Я Д А П Р Е О Б Р А З О В А Н И Я

ПРАВИЛО: Чтобы найти моду, нужно найти модальный интервал(с максимальной частотой) и воспользоваться формулой:

M = X[0]+(n[M]-n[M-1])/((n[M]-n[M-1]) + (n[M]-n[M+1]))

X[0] = 0.25 - нижняя граница модального интервала

n[M] = 97 - частота модального интервала

n[M-1] = 89 - частота предыдущего интервала

n[M+1] = 70 - частота следующего интервала

МОДА: 0.364286;

Гистограмма полученного преобразования:

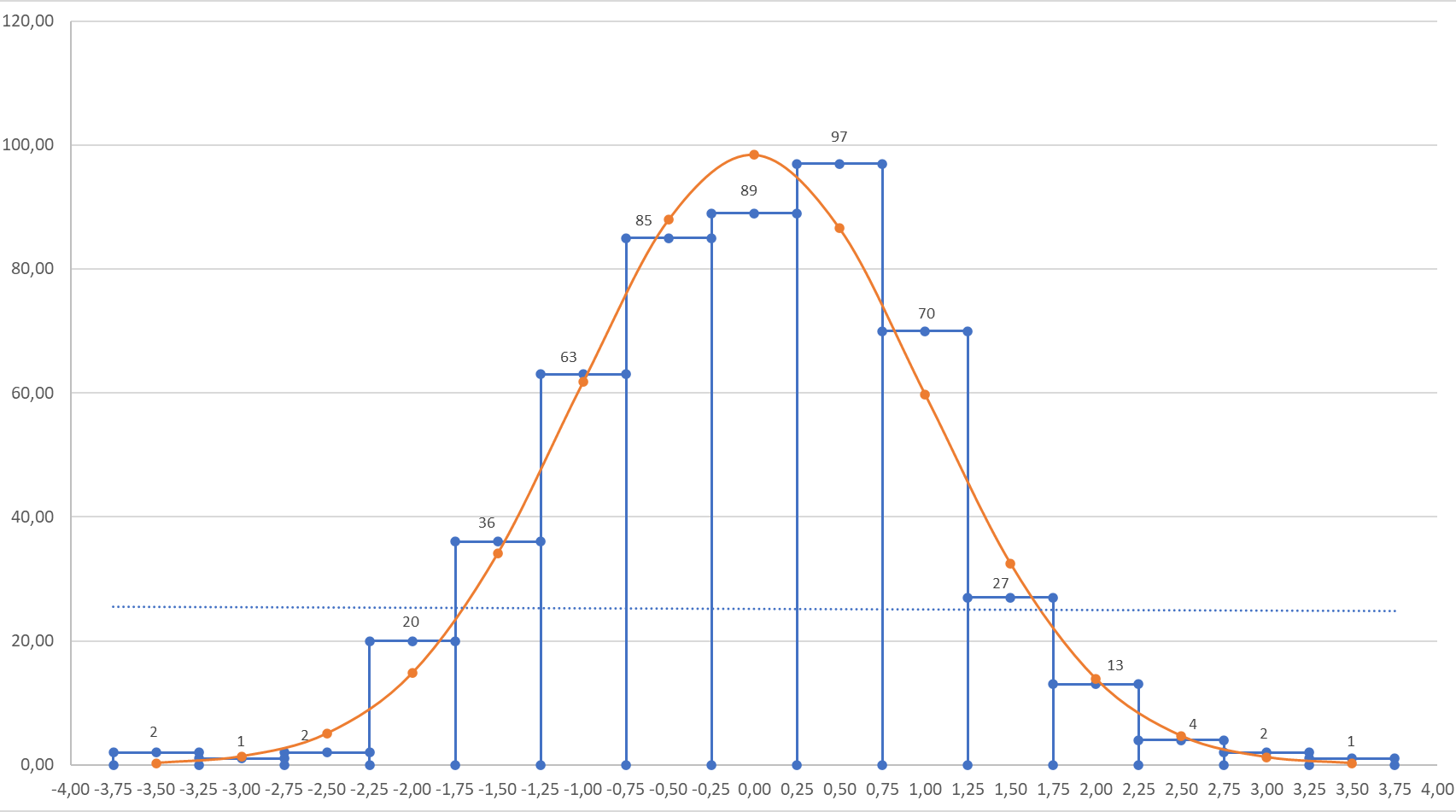
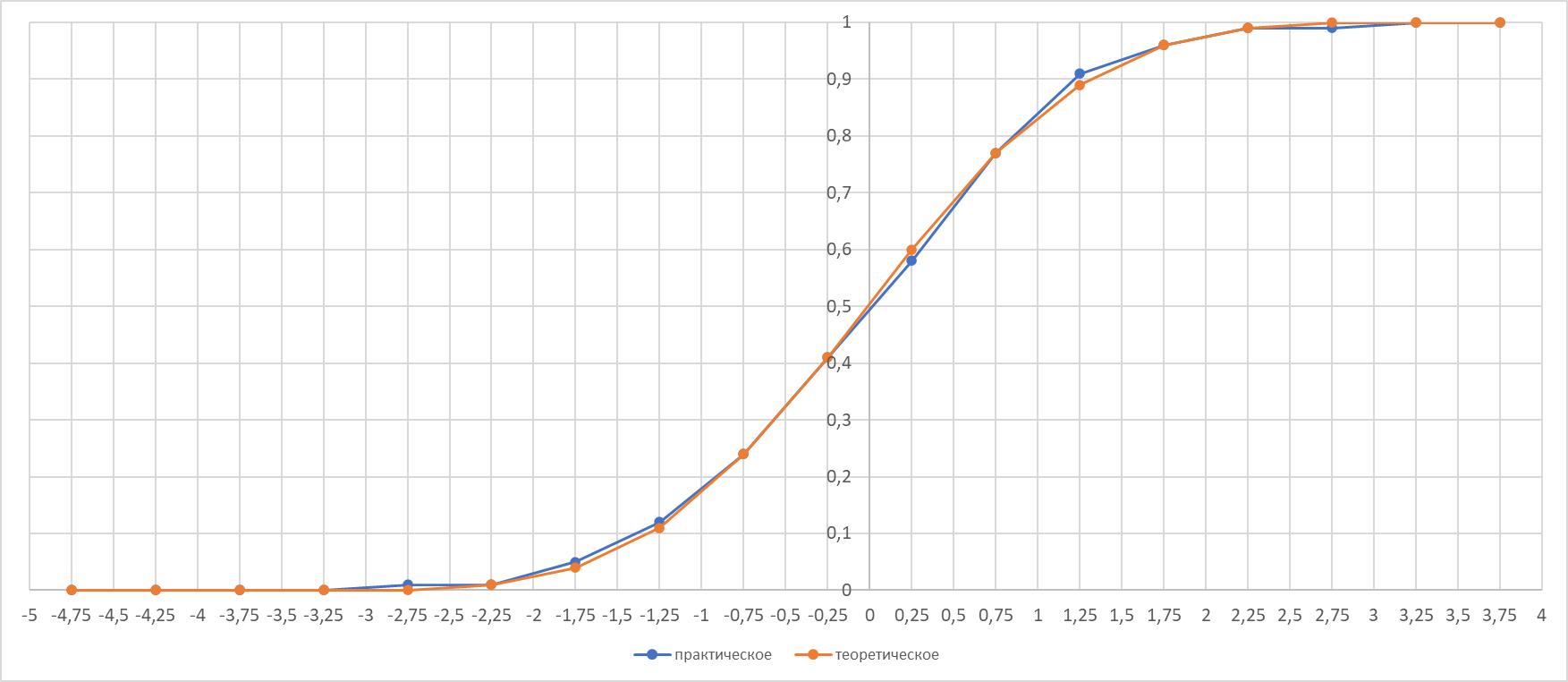


График гауссовского псевдослучайного процесса:



Гистограмма равномерно распределённой СВ fi:

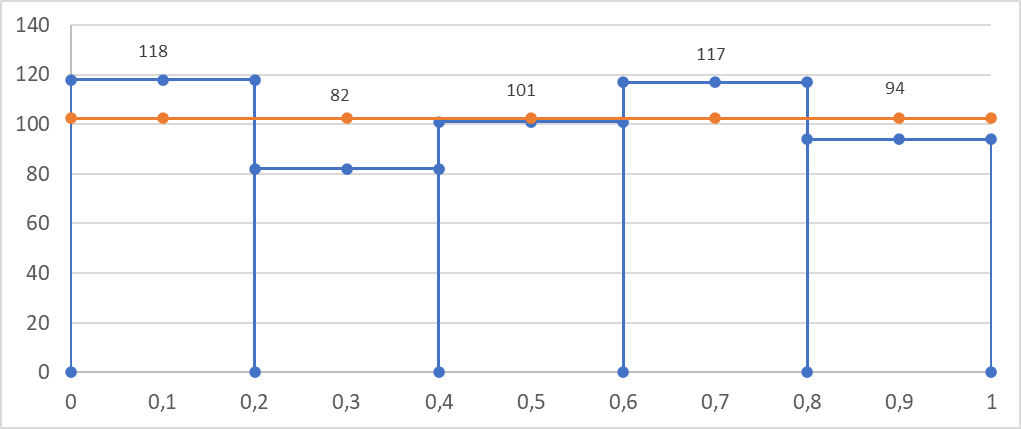
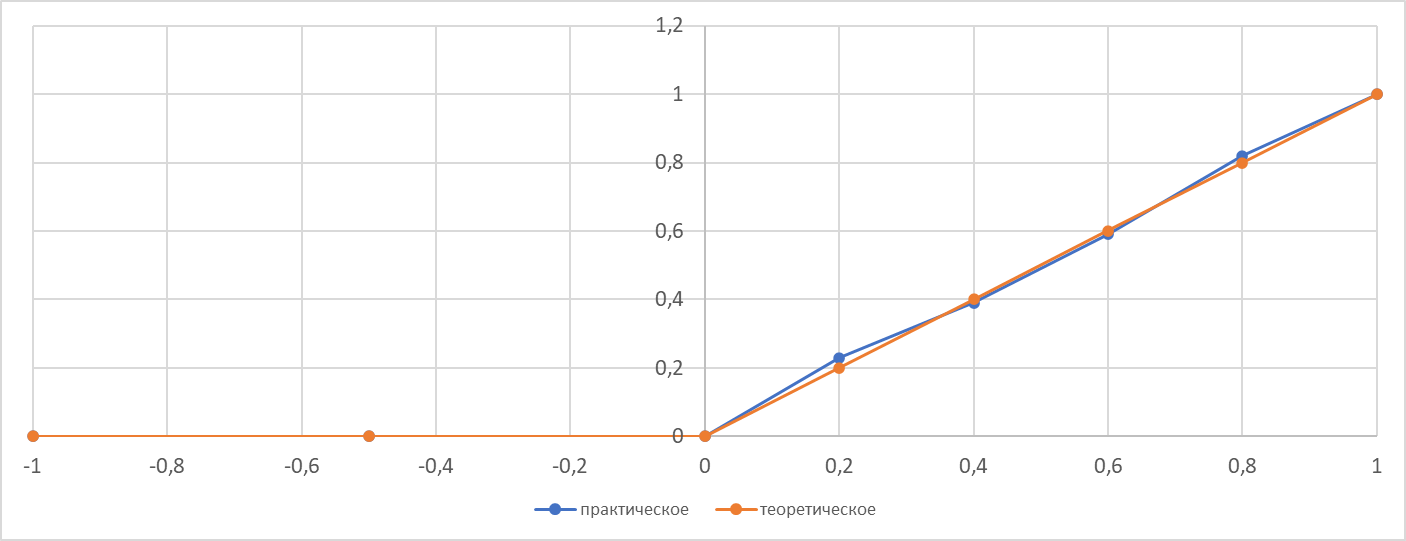


График псевдослучайного равномерного процесса СВ fi:



Гистограмма равномерно распределённой СВ R:

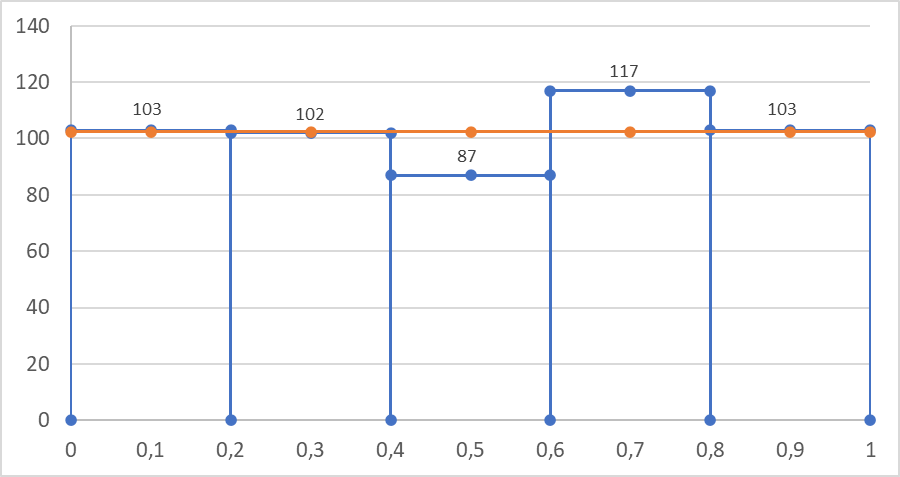
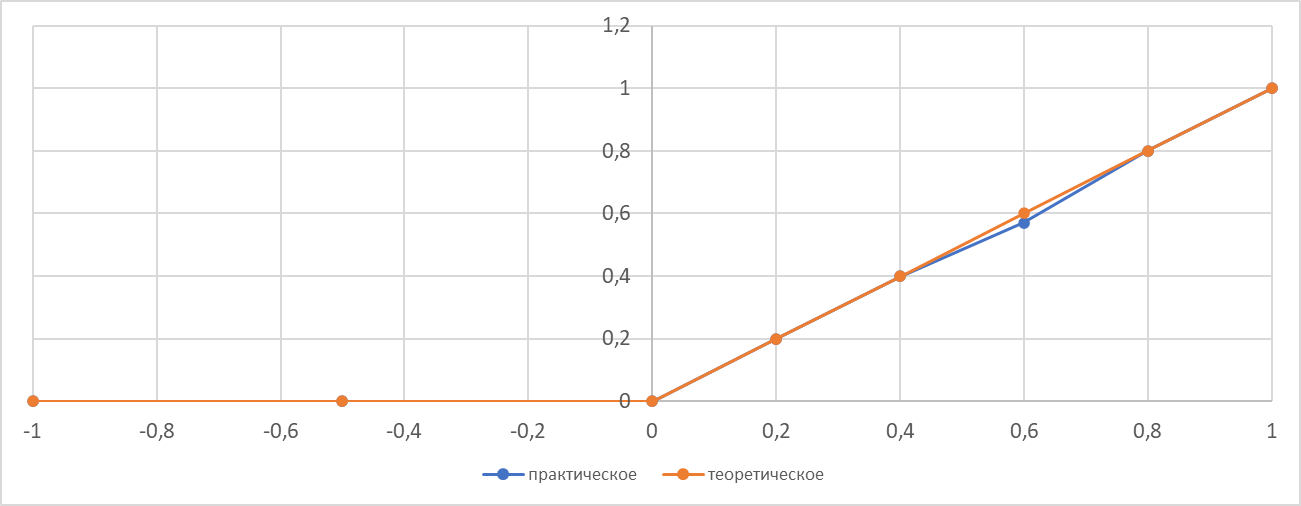


График псевдослучайного равномерного процесса СВ R:



**2.6** Цель: проверить работоспособность программы при введении количества пар псевдослучайных чисел равным 1000.

Исходные данные:

В файле считывания находится значение 1000.

Ожидаемый результат: Полученное распределение будет близким к нормальному. Исходные встроенные псевдослучайные процессы будут близки к равномерным. Результаты работы выведутся в файл и на экран.

Полученный результат:

ГАУССОВСКОЕ РАСПРЕДЕЛЕНИЕ С ПОМОЩЬЮ КОСИНУСНОГО ПРЕОБРАЗОВАНИЯ БОКСА-МЮЛЛЕРА

ЗАМЕЧАНИЕ: если Вы хотите напечатать подробно сформированную выборку, вводите количество пар не очень большим!

Количество пар псевдослучайных чисел, считанное из файла: 1000

Хотите ли Вы печатать сформированный результат преобразования и массивы равномерных процессов?

Введите Y или N латиницей : N

Вы ввели: N

Вы не захотели подробно печатать сформированные выборки

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д Л Я fi

ПРАВИЛО: Чтобы найти выборочную среднюю по первичным данным, нужно просуммировать все варианты и разделить полученный результат на объём совокупности.

Сумма вариант: 496.607;

Объём совокупности: 1000;

Выборочное среднее для данной выборки: 0.496607;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д Л Я R

Сумма вариант: 496.431;

Объём совокупности: 1000;

Выборочное среднее для данной выборки: 0.496431;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д Л Я

П Р Е О Б Р А З О В А Н И Я

Сумма вариант: -12.757;

Объём совокупности: 1000;

Выборочное среднее для данной выборки: -0.012757;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И Д Л Я fi

ОПРЕДЕЛЕНИЕ: Выборочная дисперсия - среднее арифметическое квадратов отклонений всех вариант выборки от её средней.

ПРАВИЛО: Стандартное отклонение выборки - корень из выборочной дисперсии.

Хотите ли Вы выводить подробно все квадраты отклонений? Введите Y или N латиницей:

N

Вы ввели: N

Вы не захотели подробно печатать квадраты отклонений.

Сумма квадратов отклонений: 342.685;

Выборочная дисперсия для данной выборки: 0.342685;

Стандартное отклонение: 0.585393;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И Д Л Я R

Хотите ли Вы выводить подробно все квадраты отклонений? Введите Y или N латиницей: N

Вы ввели: N

Вы не захотели подробно печатать квадраты отклонений.

Сумма квадратов отклонений: 341.273;

Выборочная дисперсия для данной выборки: 0.341273;

Стандартное отклонение: 0.584185;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И Д Л Я

П Р Е О Б Р З О В А Н И Я

Хотите ли Вы выводить подробно все квадраты отклонений? Введите Y или N латиницей: N

Вы ввели: N

Вы не захотели подробно печатать квадраты отклонений.

Сумма квадратов отклонений: 1028.82;

Выборочная дисперсия для данной выборки: 1.02882;

Стандартное отклонение: 1.01431;

Р А С Ч Ё Т М Е Д И А Н Ы В А Р И А Ц И О Н Н О Г О Р Я Д А

П Р Е О Б Р А З О В А Н И Я

ОПРЕДЕЛЕНИЕ: Медиана вариационного ряда - значение, которое делит его на две равные части (по количеству вариант).

ПРАВИЛО: Массив с выборкой должен быть отсортирован.

Количество чисел выборки 1000 - чётное число. Делим её объём пополам: 500

Берём среднее арифметическое 500 и следующего за ним элемента;

Значение медианы: -0.018902;

Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А Ц И О Н Н О Г О

Р Я Д А Д Л Я fi

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

n[i] -количество вариант, попавших в текущий интервал

ПРАВИЛО: Если варианта попадает на стык интервалов, то её следует относить в правый интервал.

Для всех интервалов, кроме последнего.

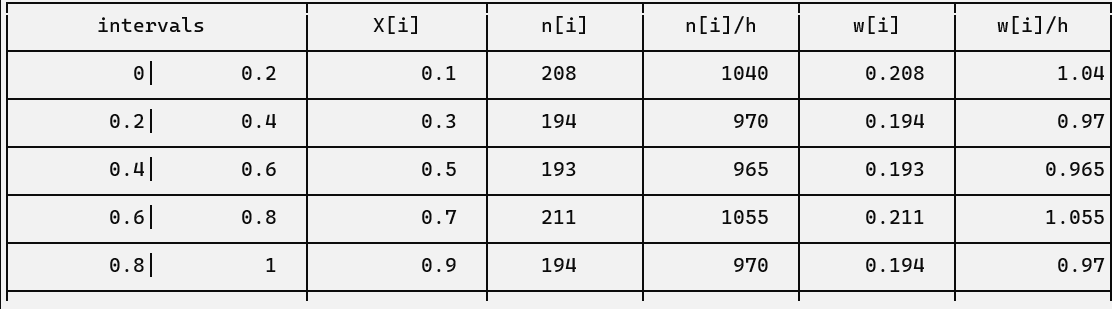
ОПРЕДЕЛЕНИЕ: Плотность частот - частота, рассчитанная на единицу ширины интервала: n[i]/h.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

САМА МАТРИЦА:



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц И И

П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я fi

Теоретические значения функции плотности равномерного распределения по формуле плотности нормального распределения

Стандартное отклонение - корень из дисперсии, sigma = 0.585393

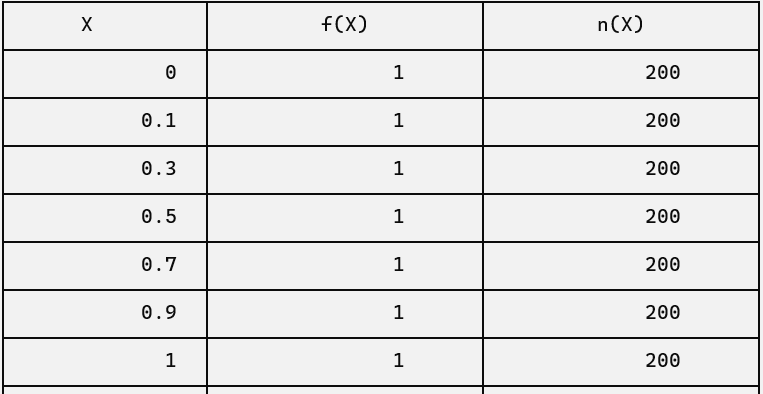
ПРАВИЛО: Теоретические частоты вычисляются в середине каждого интервала гистограммы и добавляются

вычисления в 0 и 1. Для равномерного распределения теоретическая частота вычисляется путём деления размера выборки

на количество интервалов

ПРАВИЛО: Проверку на соответствие суммы теоретических частот общему количеству элементов осуществлять суммой

значений, вычисленных только в середине интервалов



Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я fi

ПРАВИЛО: Все значения функции, что правее конца самого левого интервала в гистограмме - имеют нулевой значение функции

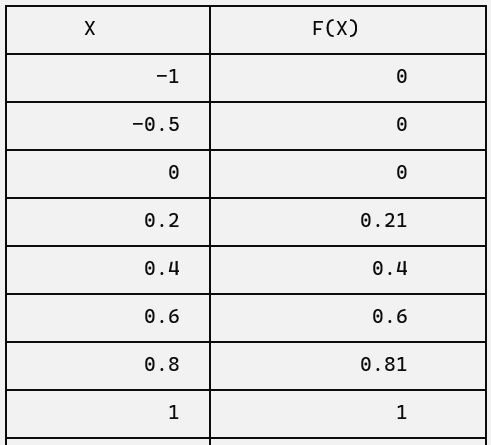
распределения далее берем крайние значения каждого интервала - за значение на оси X, накапливаем относительные частоты;

w\_n = w[i] + w\_n - значение на оси Y

Построение эмперической функции распределения

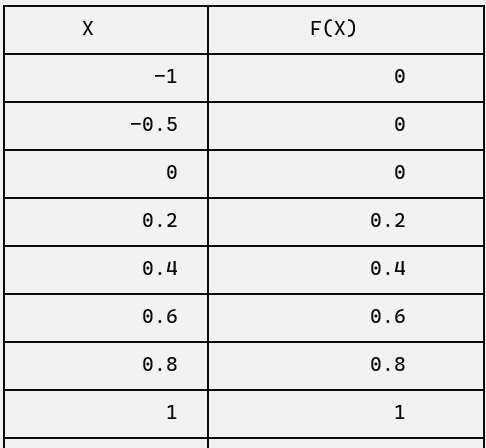
ПРАВИЛО: Значения формируются вида (x[прав]; w[i]), где w[i] -относительная частота, которая успела накопиться

на всех пройденных интервалах



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я fi

ЗАМЕЧАНИЕ: расчёт происходит по формуле функции распределения равномерного распределения в тех же точках, что и практическое;



Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А Ц И О Н Н О Г О

Р Я Д А Д Л Я R

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

n[i] -количество вариант, попавших в текущий интервал

ПРАВИЛО: Если варианта попадает на стык интервалов, то её следует относить в правый интервал.

Для всех интервалов, кроме последнего.

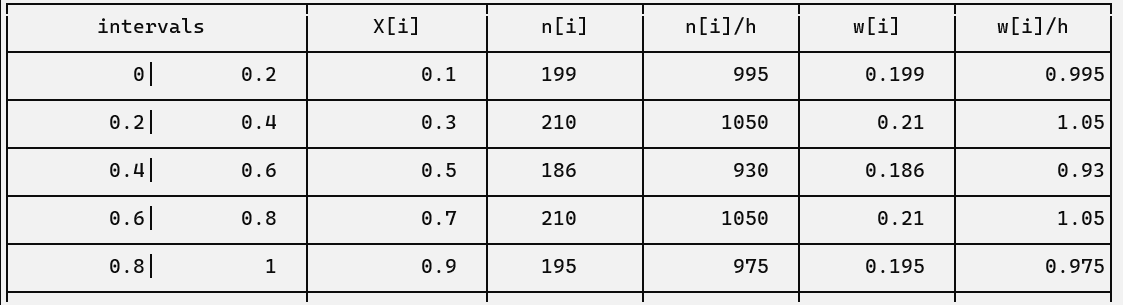
ОПРЕДЕЛЕНИЕ: Плотность частот - частота, рассчитанная на единицу ширины интервала: n[i]/h.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

САМА МАТРИЦА:



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц И И

П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я R

Теоретические значения функции плотности равномерного распределения по формуле плотности нормального распределения

Стандартное отклонение - корень из дисперсии, sigma = 0.584185

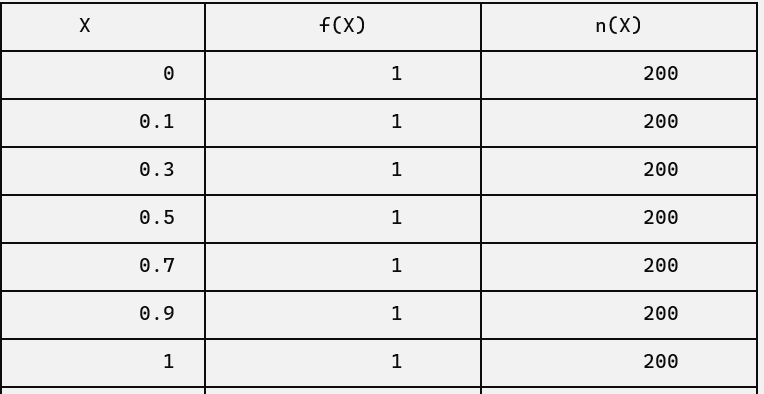
ПРАВИЛО: Теоретические частоты вычисляются в середине каждого интервала гистограммы и добавляются

вычисления в 0 и 1. Для равномерного распределения теоретическая частота вычисляется путём деления размера выборки

на количество интервалов

ПРАВИЛО: Проверку на соответствие суммы теоретических частот общему количеству элементов осуществлять суммой

значений, вычисленных только в середине интервалов



Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я R

ПРАВИЛО: Все значения функции, что правее конца самого левого интервала в гистограмме - имеют нулевой значение функции

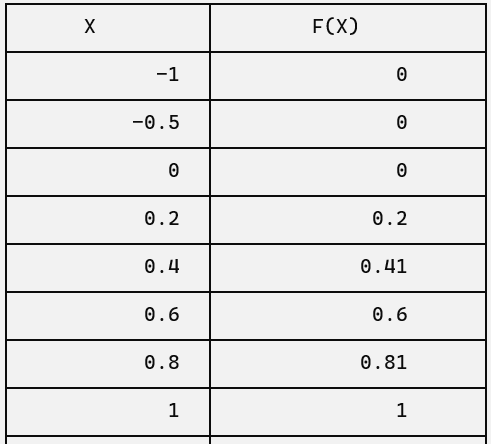
распределения алее берем крайние значения каждого интервала - за значение на оси X, накапливаем относительные частоты;

w\_n = w[i] + w\_n - значение на оси Y

Построение эмперической функции распределения

ПРАВИЛО: Значения формируются вида (x[прав]; w[i]), где w[i] -относительная частота, которая успела накопиться

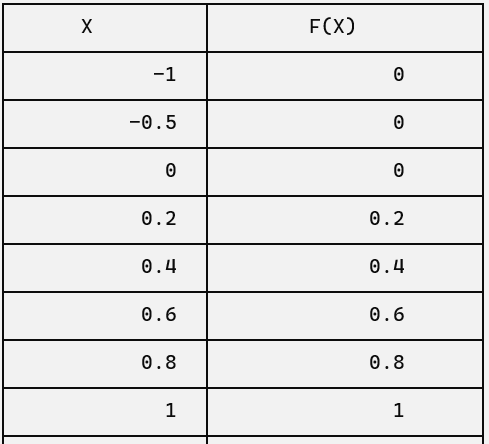
на всех пройденных интервалах



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я R

ЗАМЕЧАНИЕ: расчёт происходит по формуле функции распределения равномерного распределения в тех же точках, что и

практическое;



Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А Ц И О Н Н О Г О

Р Я Д А Д Л Я П Р Е О Б Р А З О В А Н И Я

Минимум: -3.65737

Максимум: 3.31996

X[i] - середины интервалов. Вычисляются путем вычитания из правого конца интервала левого и делением результата на 2.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

n[i] -количество вариант, попавших в текущий интервал

ПРАВИЛО: Если варианта попадает на стык интервалов, то её следует относить в правый интервал.

Для всех интервалов, кроме последнего.

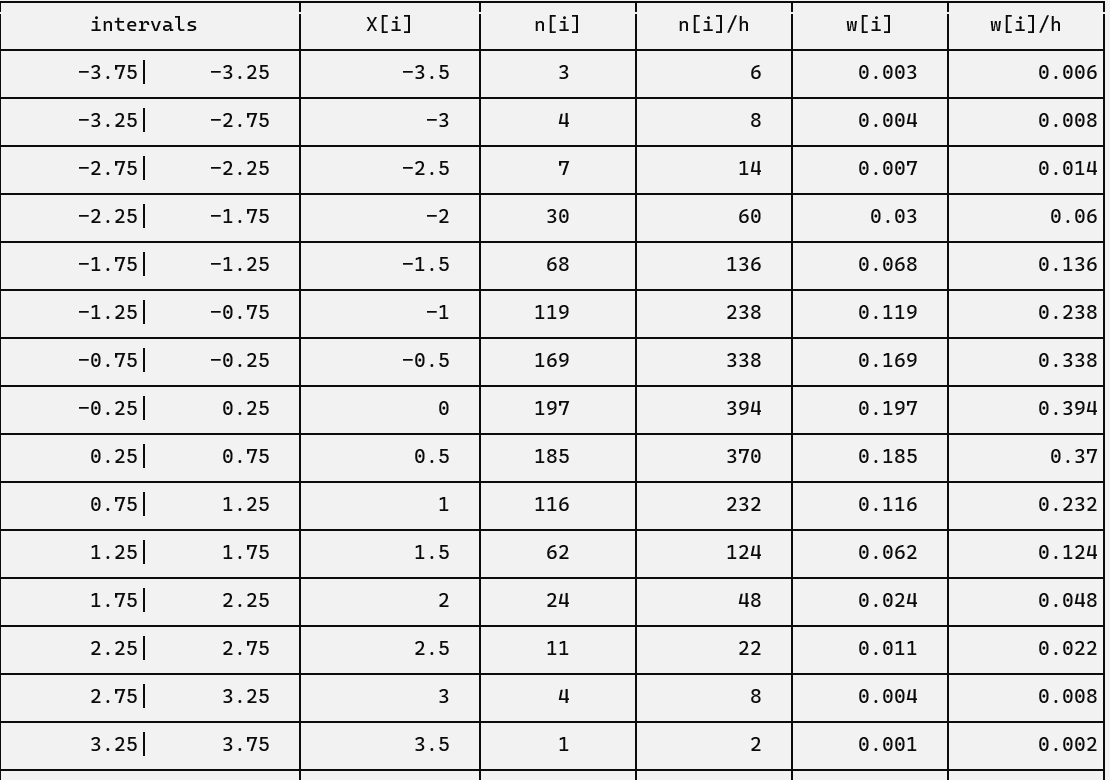
ОПРЕДЕЛЕНИЕ: Плотность частот - частота, рассчитанная на единицу ширины интервала: n[i]/h.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

САМА МАТРИЦА:



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц И И

П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я

П Р Е О Б Р А З О В А Н И Я

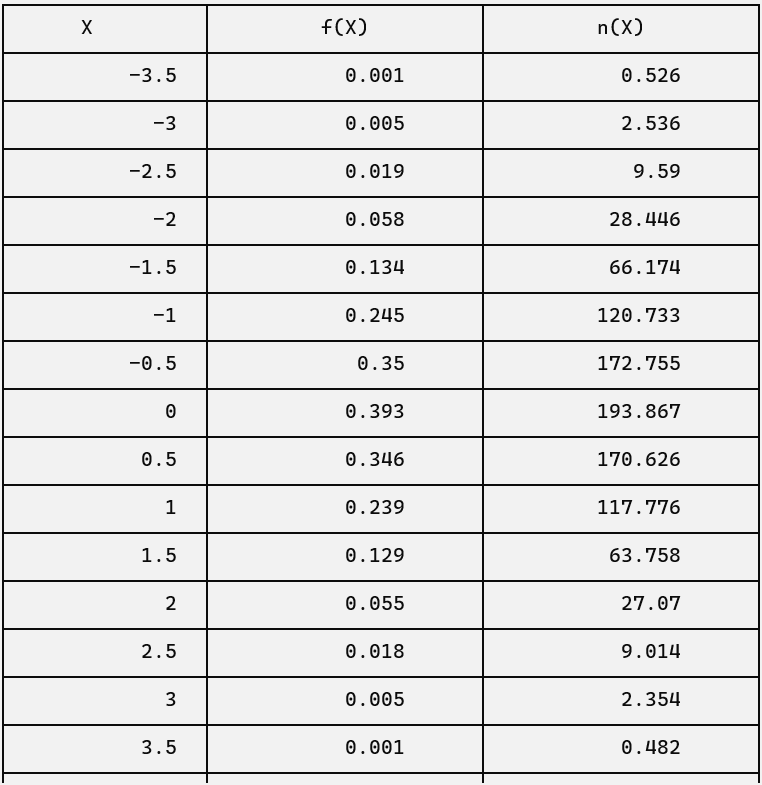
Теоретические значения функции плотности нормального распределения по вычисленному выборочному среднему и выборочной дисперсии

Стандартное отклонение - корень из дисперсии, sigma = 1.01431

ЗАМЕЧАНИЕ: Построение функции плотности распределения происходит через формулу функции плотности нормального

распределения

ПРАВИЛО: Теоретические частоты вычисляются в середине каждого интервала гистограммы



Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я

П Р Е О Б Р А З О В А Н И Я

ПРАВИЛО: Все значения функции, что правее конца самого левого интервала в гистограмме - имеют нулевой значение функции

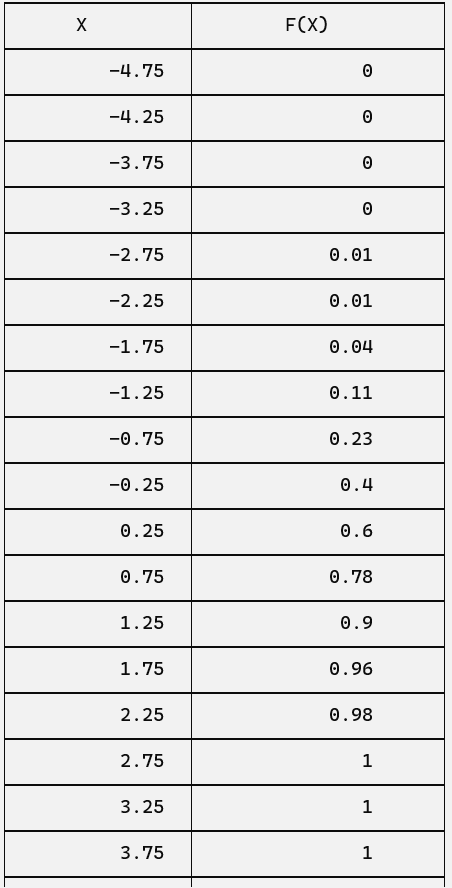
распределения алее берем крайние значения каждого интервала - за значение на оси X, накапливаем относительные частоты;

w\_n = w[i] + w\_n - значение F(X)

Построение эмперической функции распределения

ПРАВИЛО: Значения формируются вида (x[прав]; w[i]), где w[i] -относительная частота, которая успела накопиться

на всех пройденных интервалах



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я

П Р Е О Б Р А З О В А Н И Я

ЗАМЕЧАНИЕ: расчёт происходит по формуле функции распределения гауссовского распределения в тех же точках, что и

практическое;



Р А С Ч Ё Т М О Д Ы В А Р И А Ц И О Н Н О Г О Р Я Д А

П Р Е О Б Р А З О В А Н И Я

ПРАВИЛО: Чтобы найти моду, нужно найти модальный интервал(с максимальной частотой) и воспользоваться формулой:

M = X[0]+(n[M]-n[M-1])/((n[M]-n[M-1]) + (n[M]-n[M+1]))

X[0] = -0.25 - нижняя граница модального интервала

n[M] = 197 - частота модального интервала

n[M-1] = 169 - частота предыдущего интервала

n[M+1] = 185 - частота следующего интервала

МОДА: 0.1;

Гистограмма полученного преобразования:

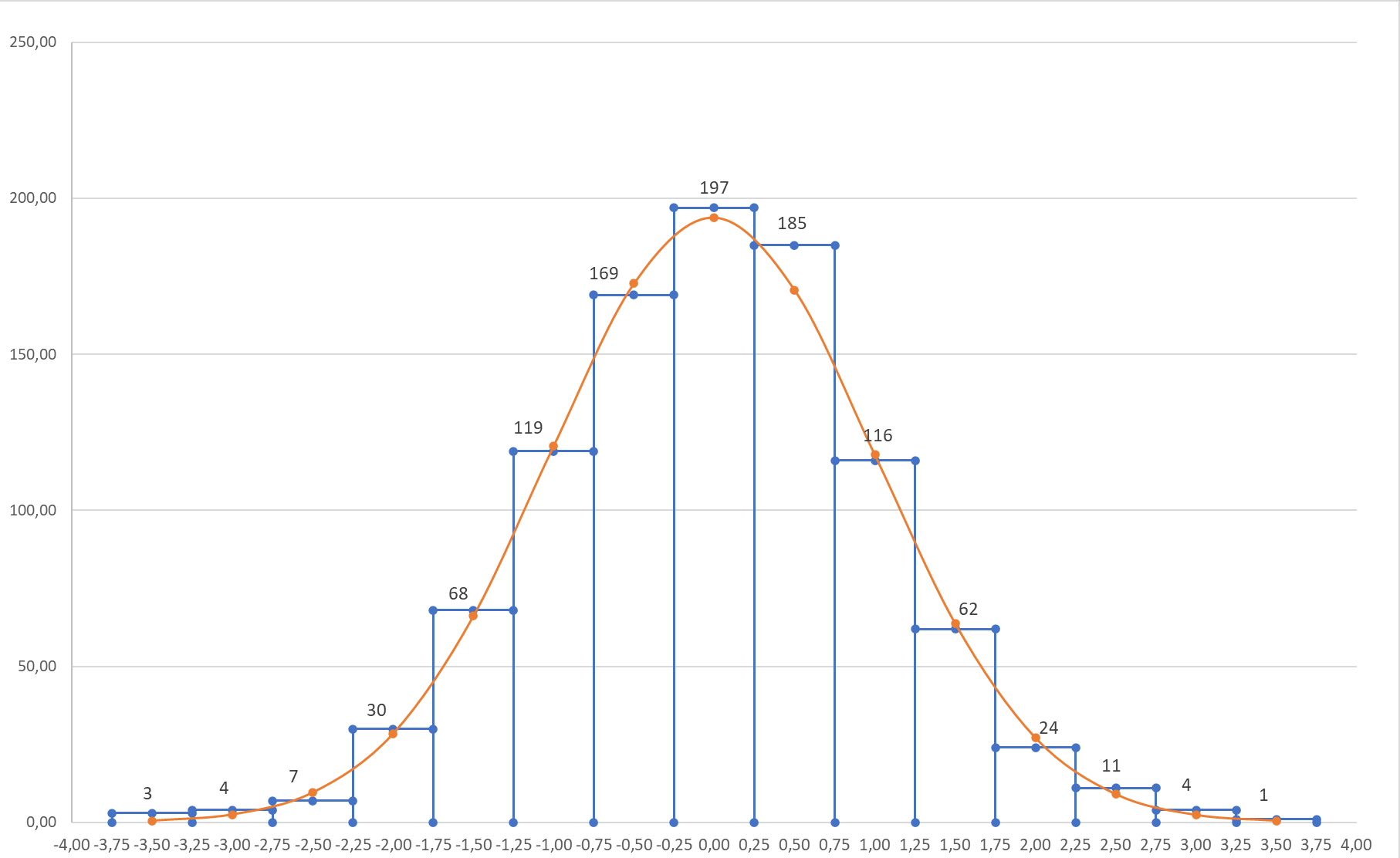
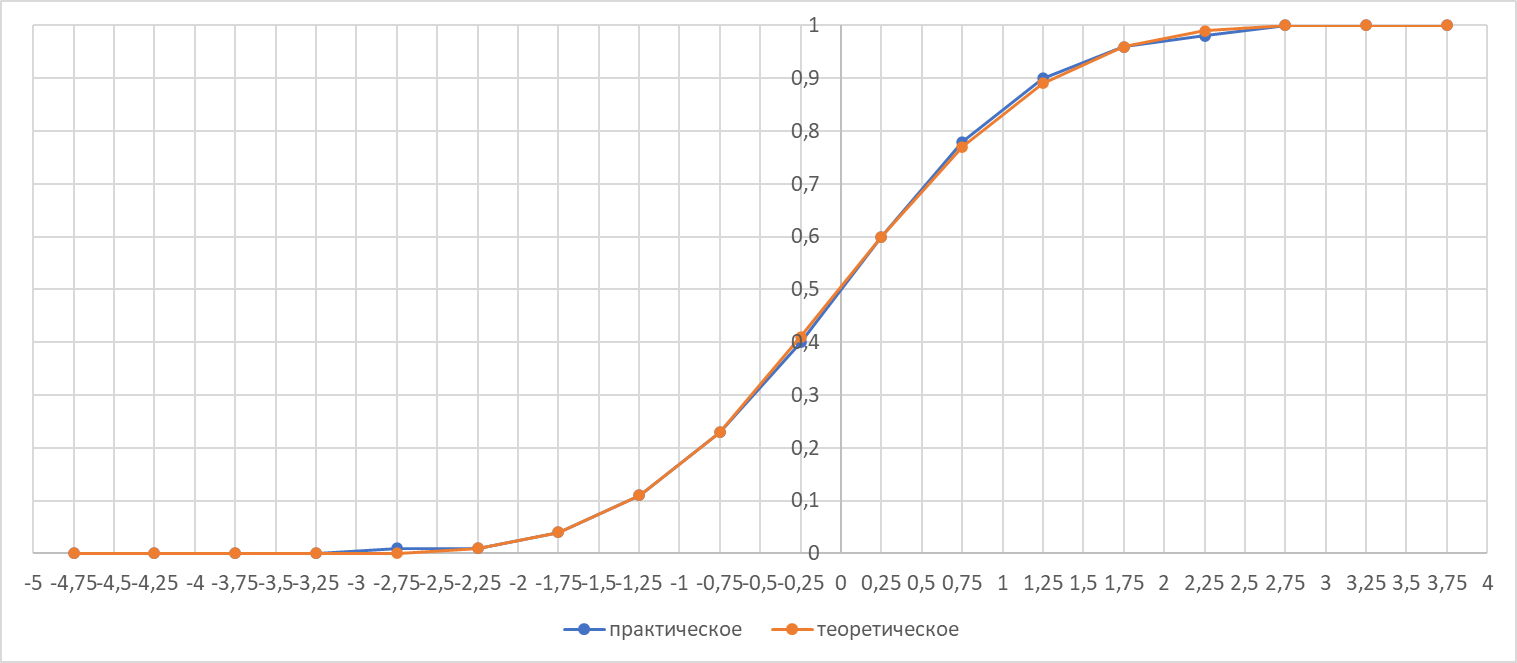


График гауссовского псевдослучайного процесса:



Гистограмма равномерно распределённой СВ fi:

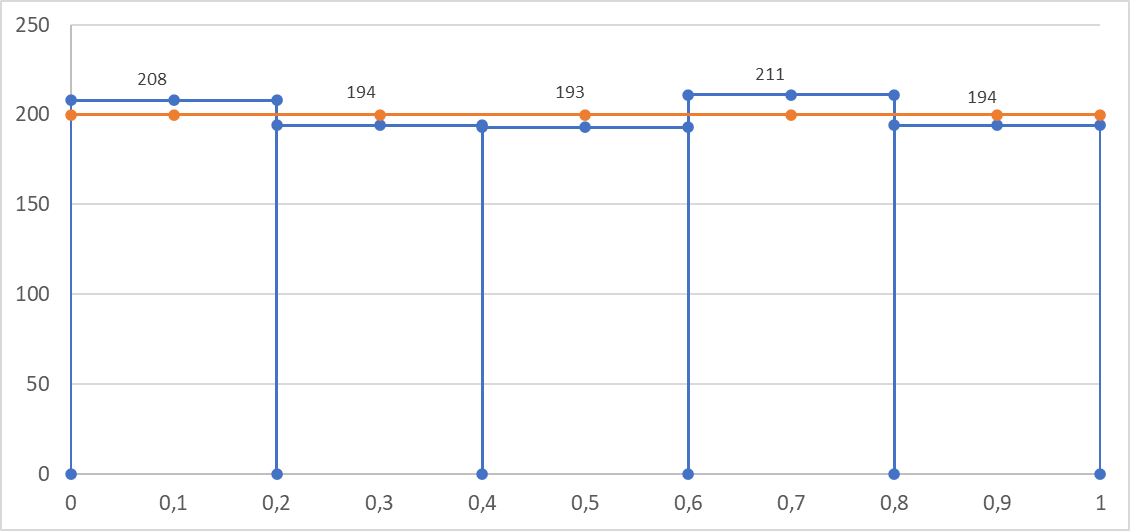
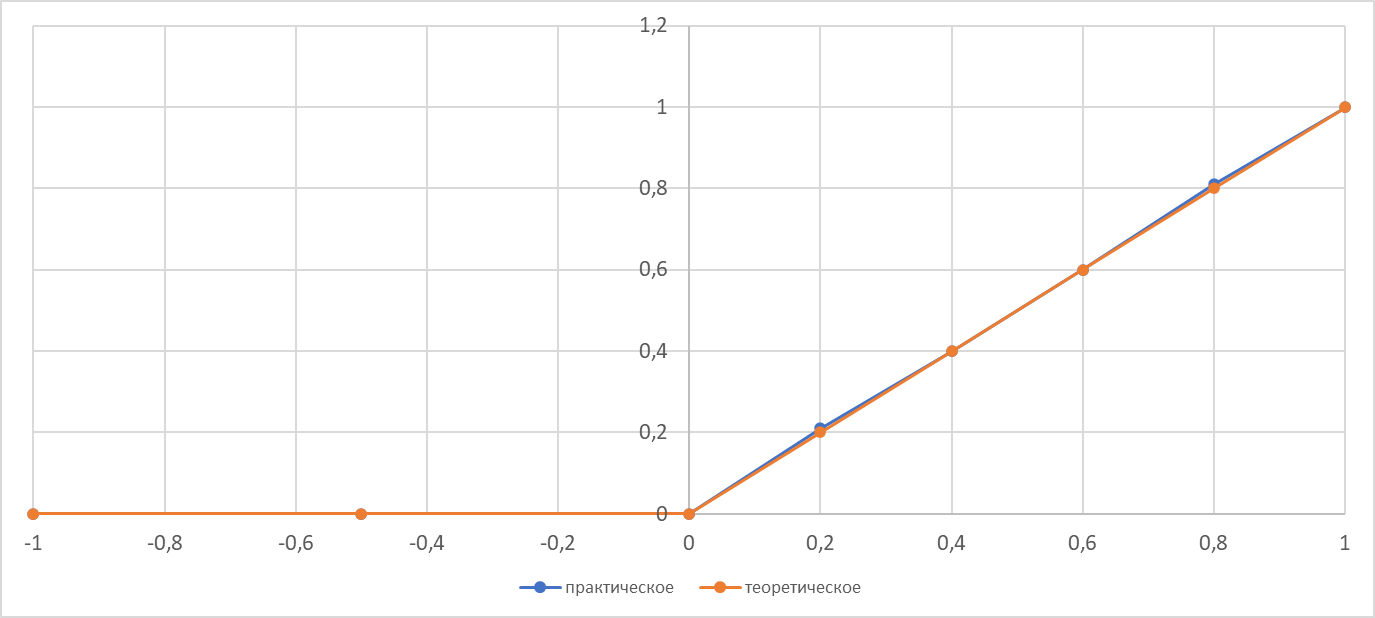


График псевдослучайного равномерного процесса СВ fi:



Гистограмма равномерно распределённой СВ R:

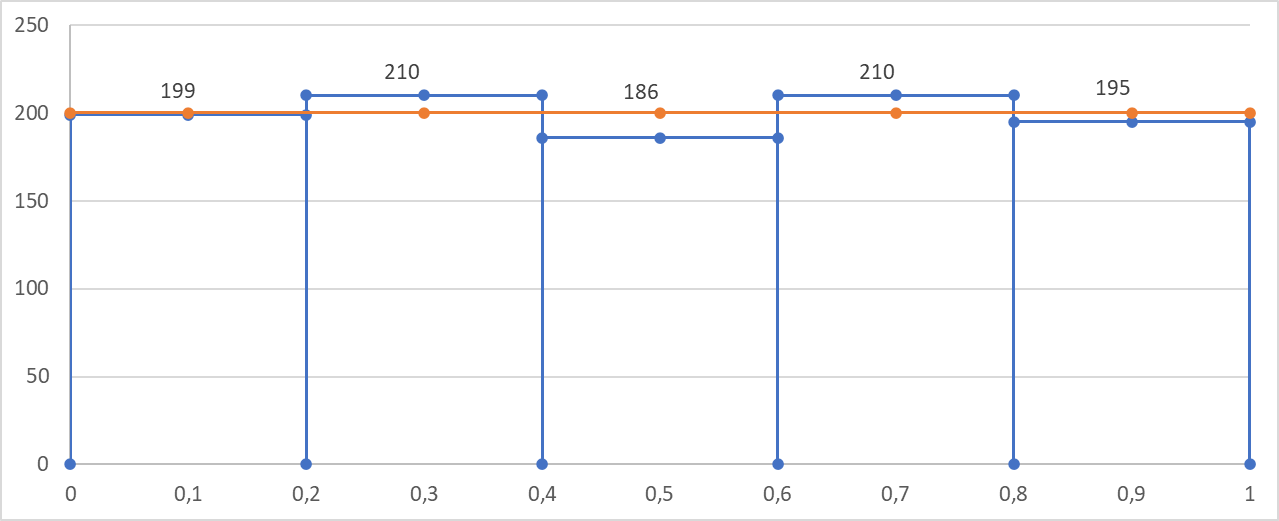
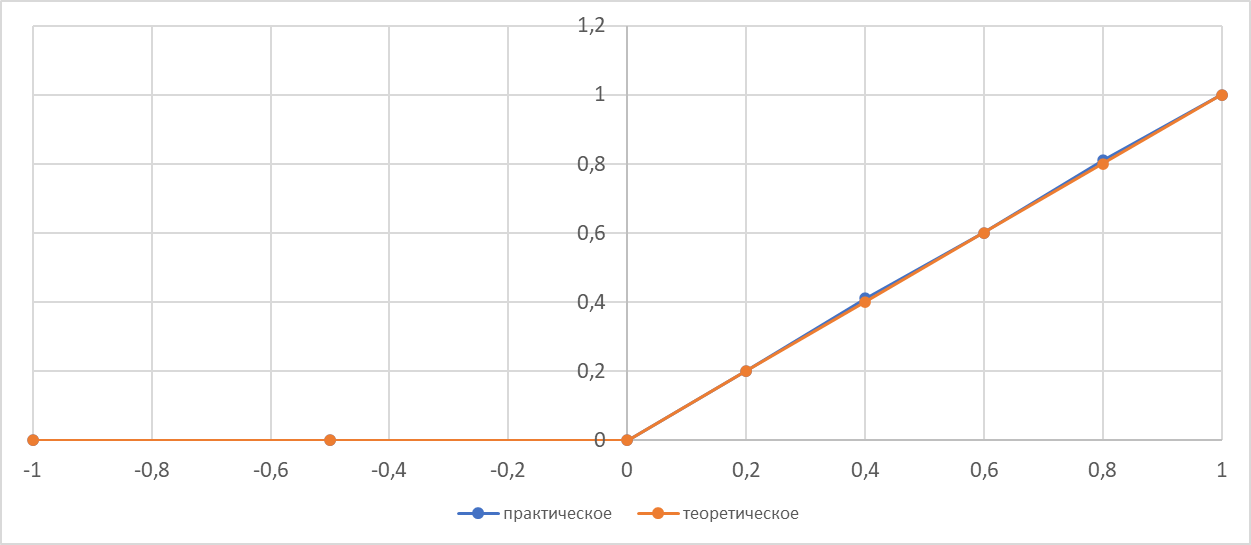


График псевдослучайного равномерного процесса СВ R:



**2.8** Цель: проверить работоспособность программы при введении количества пар псевдослучайных чисел равным 5013.

Исходные данные:

В файле считывания находится значение 5013.

Ожидаемый результат: Полученное распределение будет близким к нормальному. Исходные встроенные псевдослучайные процессы будут близки к равномерным. Результаты работы выведутся в файл и на экран.

Полученный результат:

ГАУССОВСКОЕ РАСПРЕДЕЛЕНИЕ С ПОМОЩЬЮ КОСИНУСНОГО ПРЕОБРАЗОВАНИЯ БОКСА-МЮЛЛЕРА

ЗАМЕЧАНИЕ: если Вы хотите напечатать подробно сформированную выборку, вводите количество пар не очень большим!

Количество пар псевдослучайных чисел, считанное из файла: 5013

Хотите ли Вы печатать сформированный результат преобразования и массивы равномерных процессов?

Введите Y или N латиницей : N

Вы ввели: N

Вы не захотели подробно печатать сформированные выборки

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д Л Я fi

ПРАВИЛО: Чтобы найти выборочную среднюю по первичным данным, нужно просуммировать все варианты и разделить полученный результат на объём совокупности.

Сумма вариант: 2509.53;

Объём совокупности: 5013;

Выборочное среднее для данной выборки: 0.500604;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д Л Я R

Сумма вариант: 2539.41;

Объём совокупности: 5013;

Выборочное среднее для данной выборки: 0.506564;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д Л Я

П Р Е О Б Р А З О В А Н И Я

Сумма вариант: 2.38681;

Объём совокупности: 5013;

Выборочное среднее для данной выборки: 0.000476124;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И Д Л Я fi

ОПРЕДЕЛЕНИЕ: Выборочная дисперсия - среднее арифметическое квадратов отклонений всех вариант выборки от её средней.

ПРАВИЛО: Стандартное отклонение выборки - корень из выборочной дисперсии.

Хотите ли Вы выводить подробно все квадраты отклонений? Введите Y или N латиницей: N

Вы ввели: N

Вы не захотели подробно печатать квадраты отклонений.

Сумма квадратов отклонений: 1677.36;

Выборочная дисперсия для данной выборки: 0.334601;

Стандартное отклонение: 0.578447;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И Д Л Я R

Хотите ли Вы выводить подробно все квадраты отклонений? Введите Y или N латиницей: N

Вы ввели: N

Вы не захотели подробно печатать квадраты отклонений.

Сумма квадратов отклонений: 1696.29;

Выборочная дисперсия для данной выборки: 0.338378;

Стандартное отклонение: 0.581702;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И Д Л Я

П Р Е О Б Р З О В А Н И Я

Хотите ли Вы выводить подробно все квадраты отклонений? Введите Y или N латиницей: N

Вы ввели: N

Вы не захотели подробно печатать квадраты отклонений.

Сумма квадратов отклонений: 4988.39;

Выборочная дисперсия для данной выборки: 0.995091;

Стандартное отклонение: 0.997543;

Р А С Ч Ё Т М Е Д И А Н Ы В А Р И А Ц И О Н Н О Г О Р Я Д А

П Р Е О Б Р А З О В А Н И Я

ОПРЕДЕЛЕНИЕ: Медиана вариационного ряда - значение, которое делит его на две равные части (по количеству вариант).

ПРАВИЛО: Массив с выборкой должен быть отсортирован.

Количество чисел выборки 5013 - нечётное число. Делим её объём пополам и округляем в большую сторону

Индекс медианы выборки: 2506;

Значение медианы: -0.0149057;

Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А Ц И О Н Н О Г О

Р Я Д А Д Л Я fi

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

n[i] -количество вариант, попавших в текущий интервал

ПРАВИЛО: Если варианта попадает на стык интервалов, то её следует относить в правый интервал.

Для всех интервалов, кроме последнего.

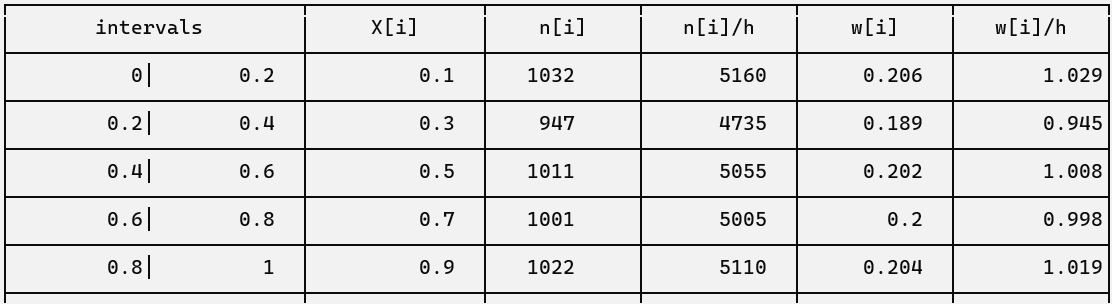
ОПРЕДЕЛЕНИЕ: Плотность частот - частота, рассчитанная на единицу ширины интервала: n[i]/h.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

САМА МАТРИЦА:



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц И И

П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я fi

Теоретические значения функции плотности равномерного распределения по формуле плотности нормального распределения

Стандартное отклонение - корень из дисперсии, sigma = 0.578447

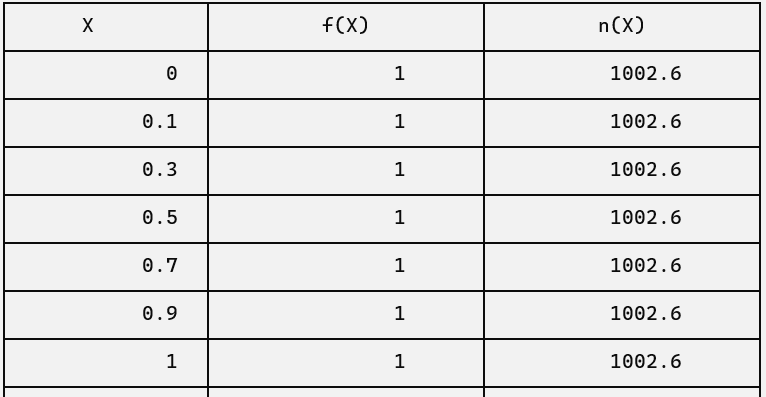
ПРАВИЛО: Теоретические частоты вычисляются в середине каждого интервала гистограммы и добавляются

вычисления в 0 и 1. Для равномерного распределения теоретическая частота вычисляется путём деления размера выборки

на количество интервалов

ПРАВИЛО: Проверку на соответствие суммы теоретических частот общему количеству элементов осуществлять суммой

значений, вычисленных только в середине интервалов



Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я fi

ПРАВИЛО: Все значения функции, что правее конца самого левого интервала в гистограмме - имеют нулевой значение функции

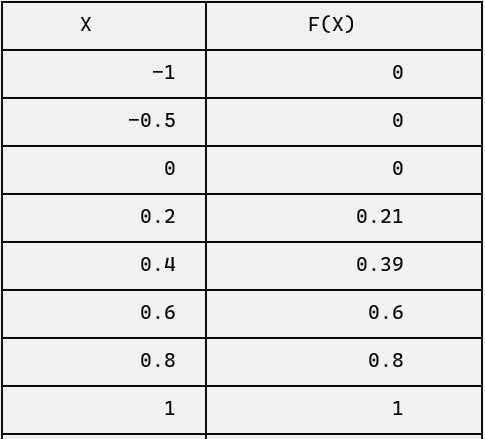
распределения далее берем крайние значения каждого интервала - за значение на оси X, накапливаем относительные частоты;

w\_n = w[i] + w\_n - значение на оси Y

Построение эмперической функции распределения

ПРАВИЛО: Значения формируются вида (x[прав]; w[i]), где w[i] -относительная частота, которая успела накопиться

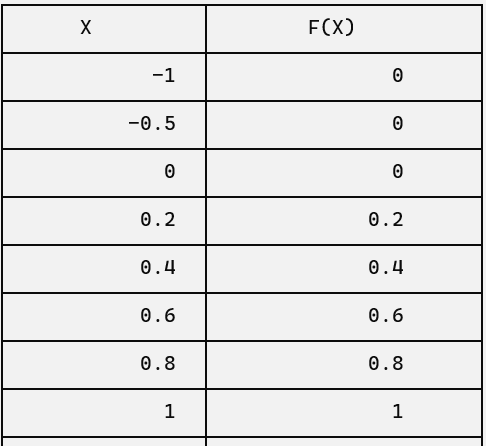
на всех пройденных интервалах



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я fi

ЗАМЕЧАНИЕ: расчёт происходит по формуле функции распределения равномерного распределения в тех же точках, что и

практическое;



Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А Ц И О Н Н О Г О

Р Я Д А Д Л Я R

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

n[i] -количество вариант, попавших в текущий интервал

ПРАВИЛО: Если варианта попадает на стык интервалов, то её следует относить в правый интервал.

Для всех интервалов, кроме последнего.

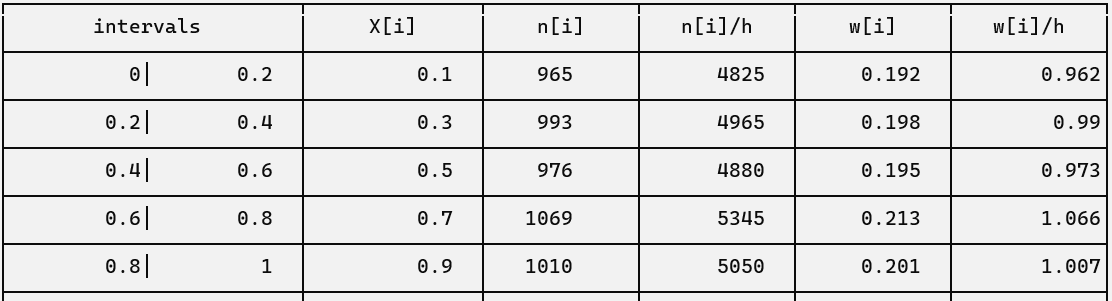
ОПРЕДЕЛЕНИЕ: Плотность частот - частота, рассчитанная на единицу ширины интервала: n[i]/h.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

САМА МАТРИЦА:



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц И И

П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я R

Теоретические значения функции плотности равномерного распределения по формуле плотности нормального распределения

Стандартное отклонение - корень из дисперсии, sigma = 0.581702

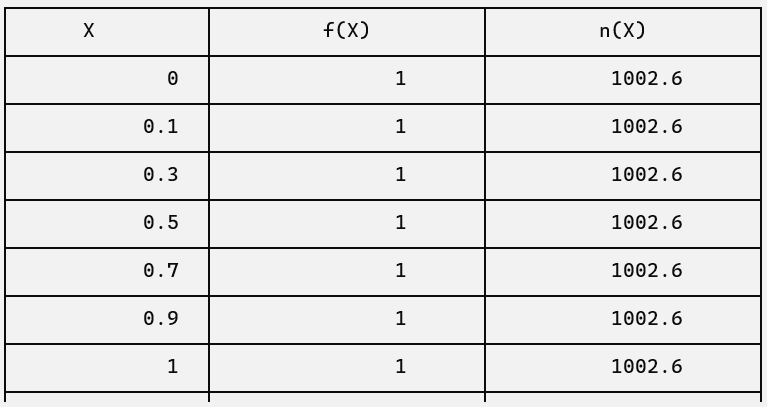
ПРАВИЛО: Теоретические частоты вычисляются в середине каждого интервала гистограммы и добавляются

вычисления в 0 и 1. Для равномерного распределения теоретическая частота вычисляется путём деления размера выборки

на количество интервалов

ПРАВИЛО: Проверку на соответствие суммы теоретических частот общему количеству элементов осуществлять суммой

значений, вычисленных только в середине интервалов



Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я R

ПРАВИЛО: Все значения функции, что правее конца самого левого интервала в гистограмме - имеют нулевой значение функции

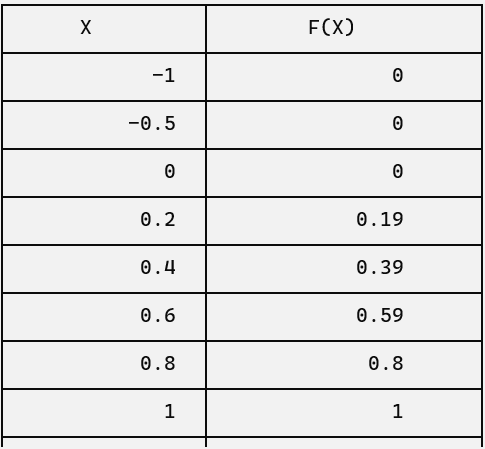
распределения алее берем крайние значения каждого интервала - за значение на оси X, накапливаем относительные частоты;

w\_n = w[i] + w\_n - значение на оси Y

Построение эмперической функции распределения

ПРАВИЛО: Значения формируются вида (x[прав]; w[i]), где w[i] -относительная частота, которая успела накопиться

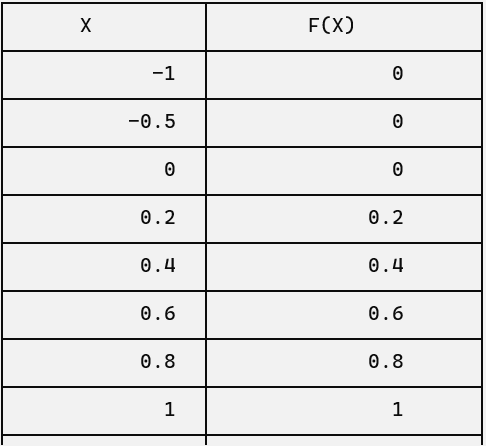
на всех пройденных интервалах



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я R

ЗАМЕЧАНИЕ: расчёт происходит по формуле функции распределения равномерного распределения в тех же точках, что и

практическое;



Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А Ц И О Н Н О Г О

Р Я Д А Д Л Я П Р Е О Б Р А З О В А Н И Я

Минимум: -3.91863

Максимум: 3.40506

X[i] - середины интервалов. Вычисляются путем вычитания из правого конца интервала левого и делением результата на 2.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

n[i] -количество вариант, попавших в текущий интервал

ПРАВИЛО: Если варианта попадает на стык интервалов, то её следует относить в правый интервал.

Для всех интервалов, кроме последнего.

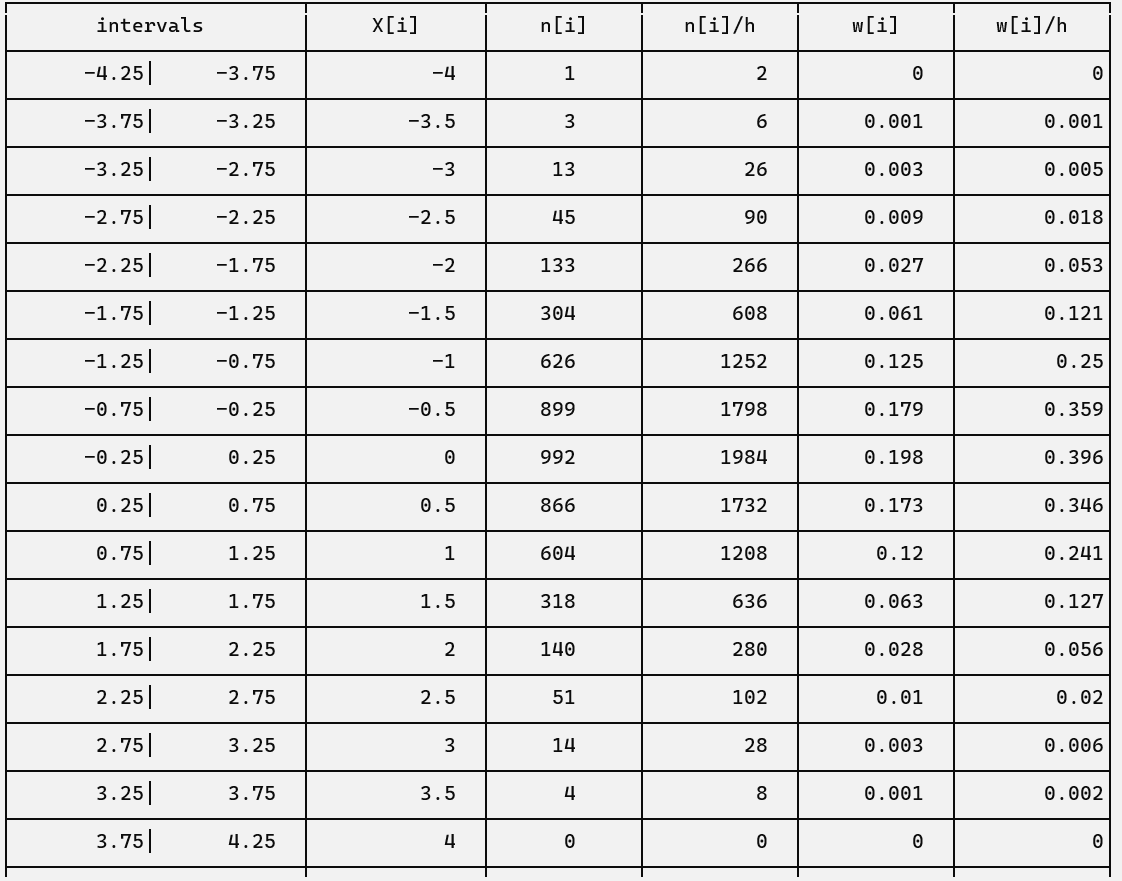
ОПРЕДЕЛЕНИЕ: Плотность частот - частота, рассчитанная на единицу ширины интервала: n[i]/h.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

САМА МАТРИЦА:



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц И И

П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я

П Р Е О Б Р А З О В А Н И Я

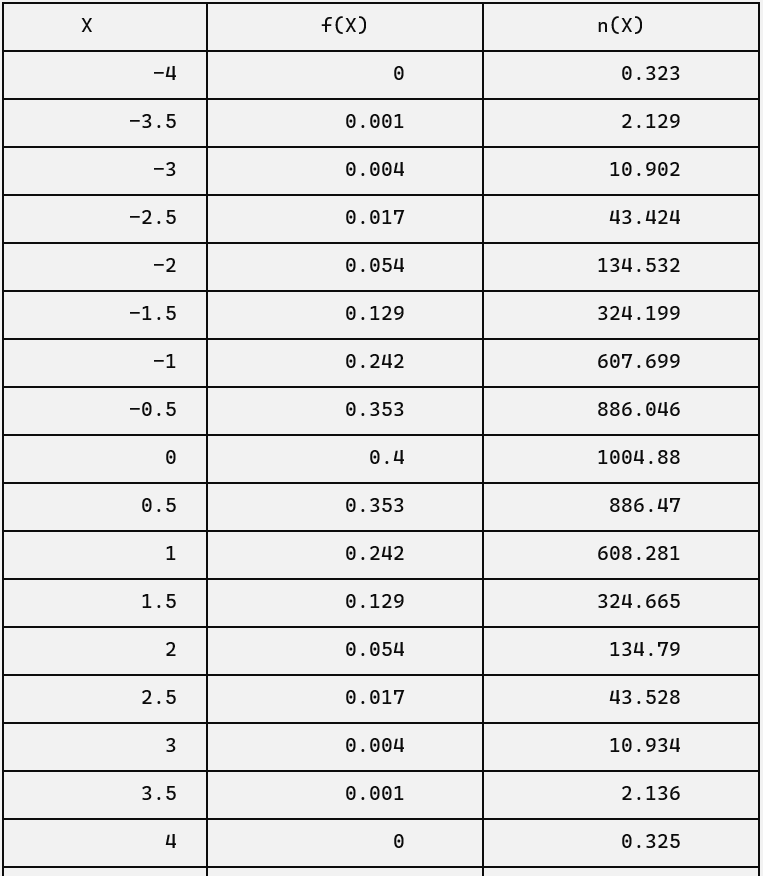
Теоретические значения функции плотности нормального распределения по вычисленному выборочному среднему и выборочной дисперсии

Стандартное отклонение - корень из дисперсии, sigma = 0.997543

ЗАМЕЧАНИЕ: Построение функции плотности распределения происходит через формулу функции плотности нормального

распределения

ПРАВИЛО: Теоретические частоты вычисляются в середине каждого интервала гистограммы



Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я

П Р Е О Б Р А З О В А Н И Я

ПРАВИЛО: Все значения функции, что правее конца самого левого интервала в гистограмме - имеют нулевой значение функции

распределения алее берем крайние значения каждого интервала - за значение на оси X, накапливаем относительные частоты;

w\_n = w[i] + w\_n - значение F(X)

Построение эмперической функции распределения

ПРАВИЛО: Значения формируются вида (x[прав]; w[i]), где w[i] -относительная частота, которая успела накопиться

на всех пройденных интервалах

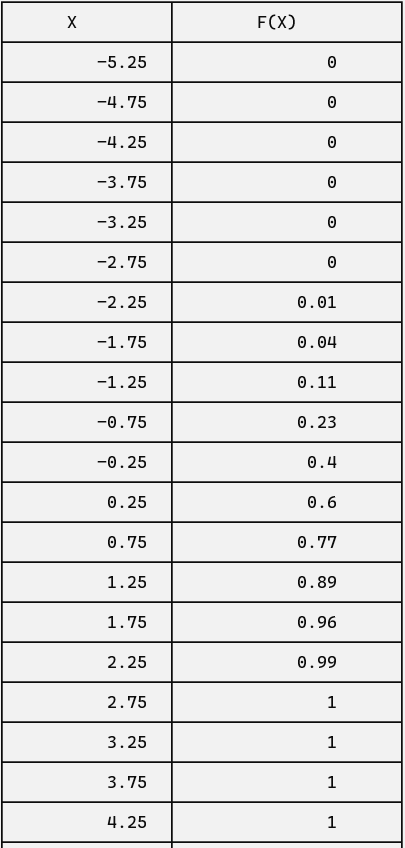


Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я

П Р Е О Б Р А З О В А Н И Я

ЗАМЕЧАНИЕ: расчёт происходит по формуле функции распределения гауссовского распределения в тех же точках, что и

практическое;



Р А С Ч Ё Т М О Д Ы В А Р И А Ц И О Н Н О Г О Р Я Д А

П Р Е О Б Р А З О В А Н И Я

ПРАВИЛО: Чтобы найти моду, нужно найти модальный интервал(с максимальной частотой) и воспользоваться формулой:

M = X[0]+(n[M]-n[M-1])/((n[M]-n[M-1]) + (n[M]-n[M+1]))

X[0] = -0.25 - нижняя граница модального интервала

n[M] = 992 - частота модального интервала

n[M-1] = 899 - частота предыдущего интервала

n[M+1] = 866 - частота следующего интервала

МОДА: -0.0376712;

Гистограмма полученного преобразования:

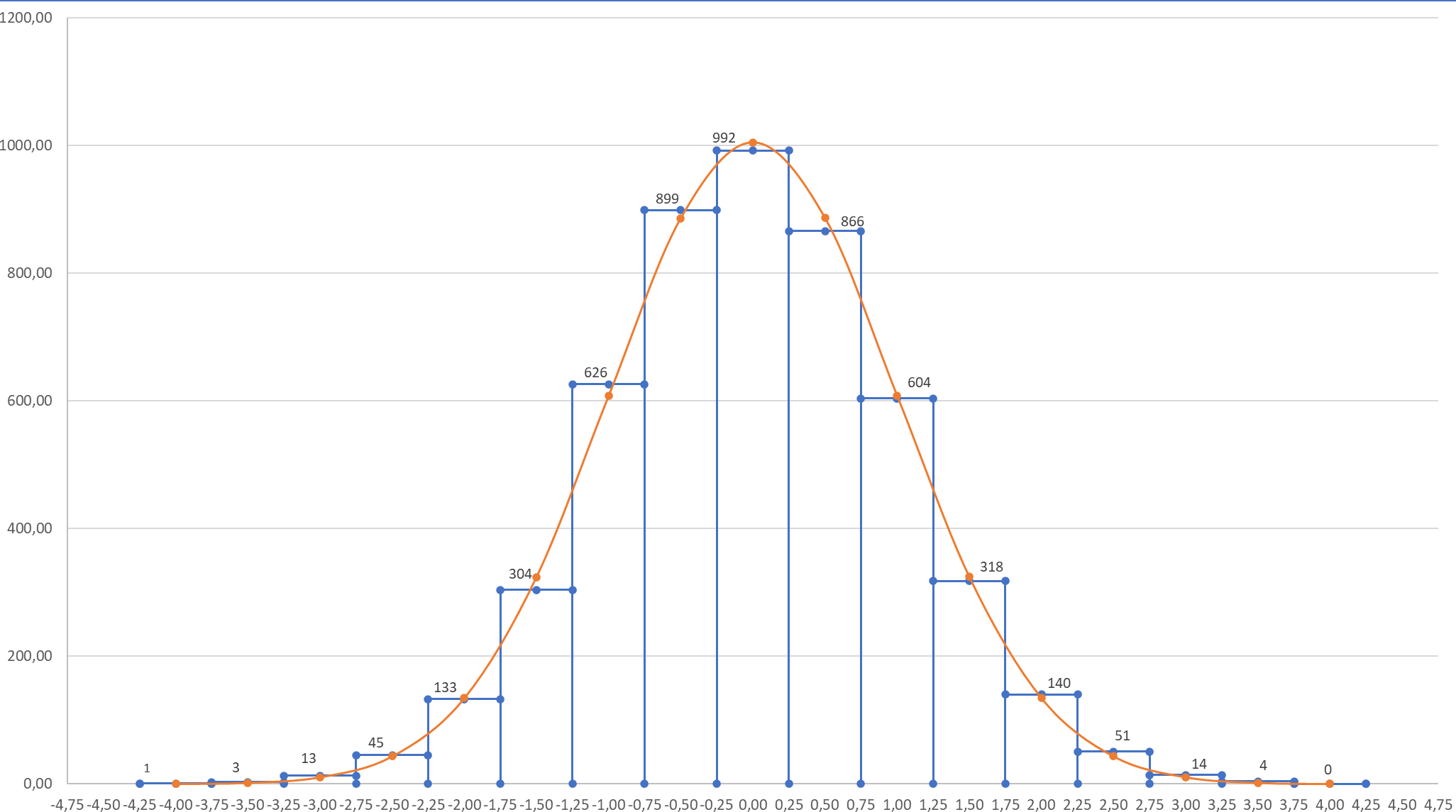
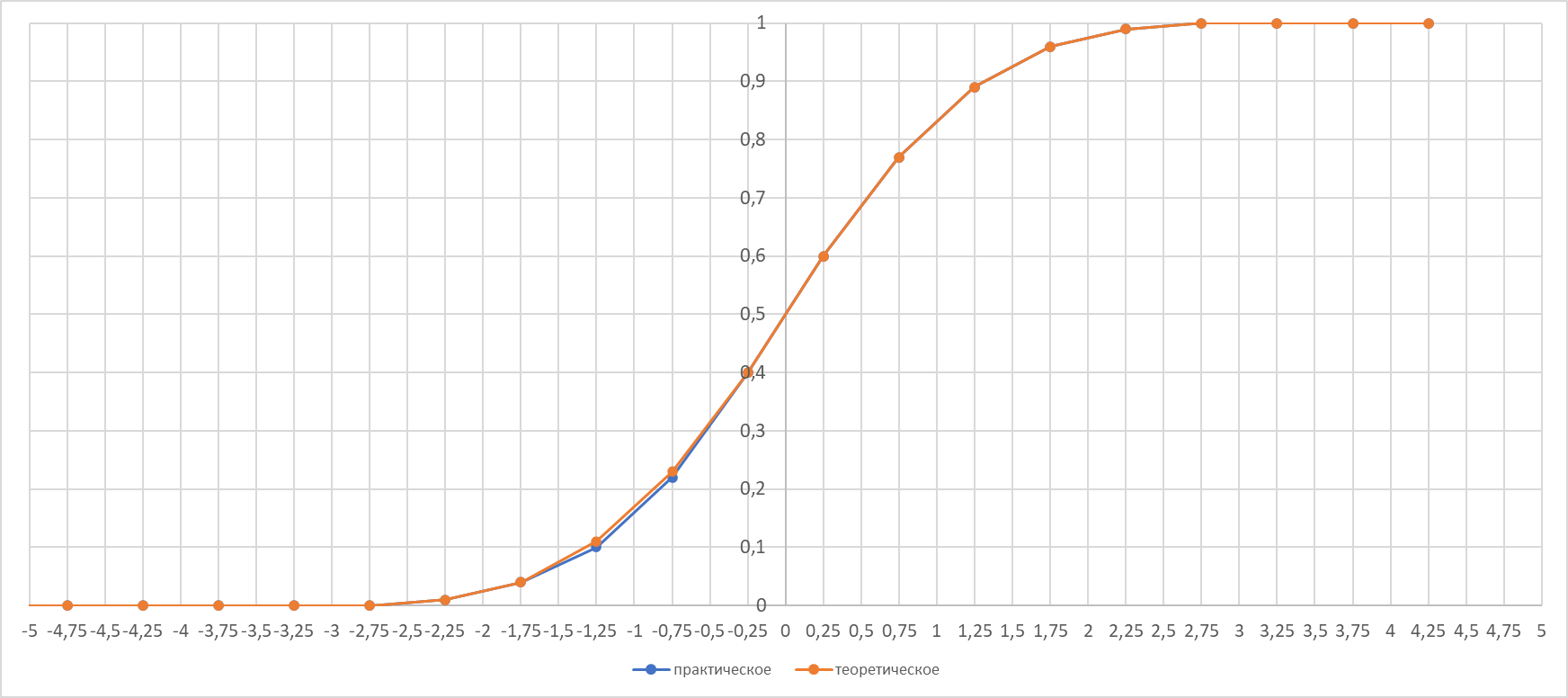


График гауссовского псевдослучайного процесса:



Гистограмма равномерно распределённой СВ fi:

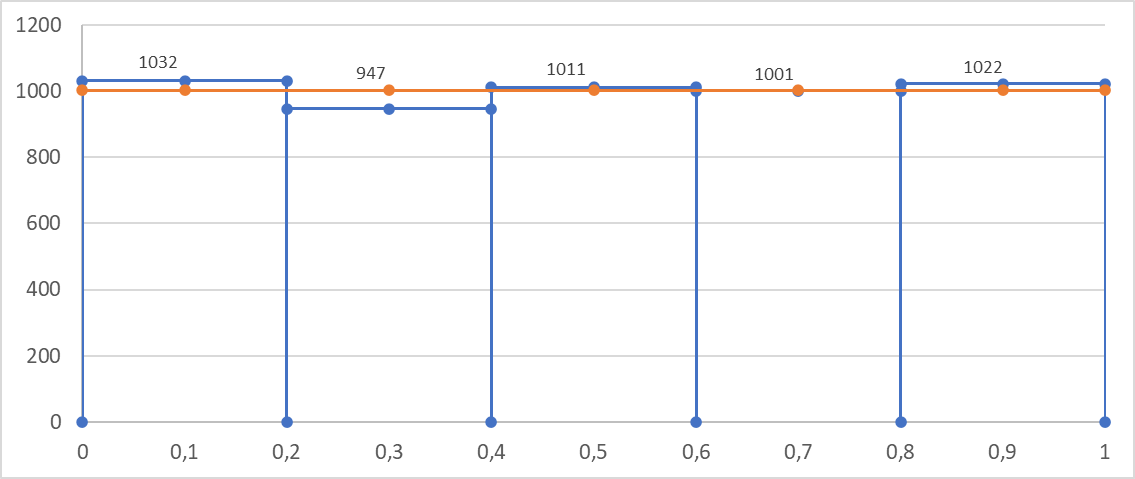
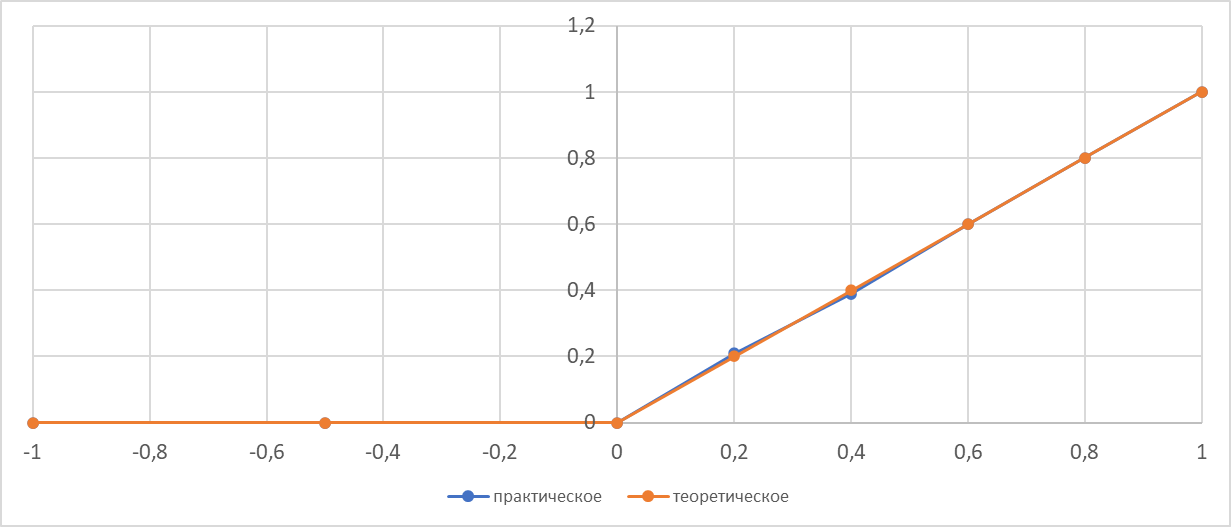


График псевдослучайного равномерного процесса СВ fi:



Гистограмма равномерно распределённой СВ R:

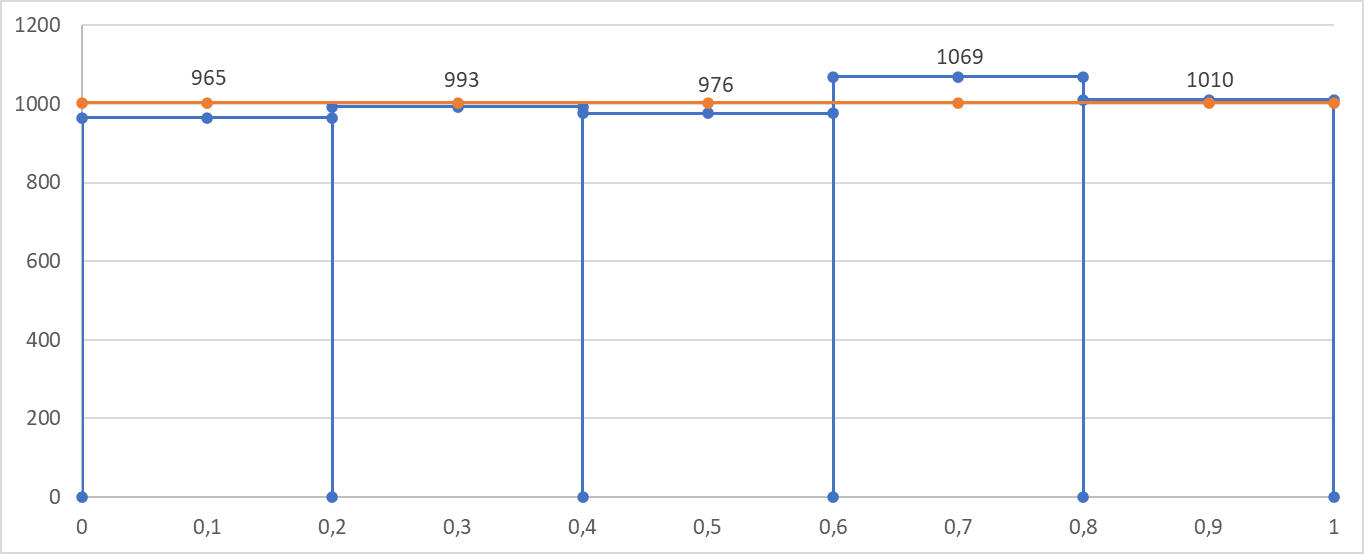
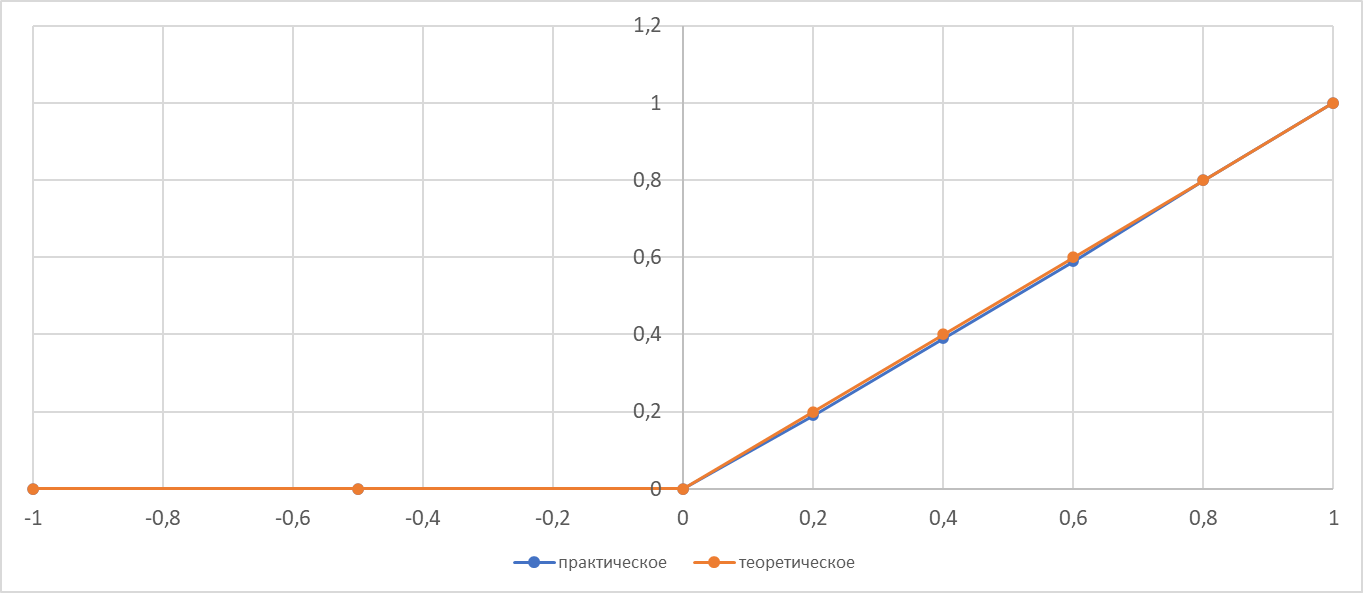


График псевдослучайного равномерного процесса СВ R:



**2.9** Цель: проверить работоспособность программы при введении количества пар псевдослучайных чисел равным 10000.

Исходные данные:

В файле считывания находится значение 10000.

Ожидаемый результат: Полученное распределение будет близким к нормальному. Исходные встроенные псевдослучайные процессы будут близки к равномерным. Результаты работы выведутся в файл и на экран.

Полученный результат:

ГАУССОВСКОЕ РАСПРЕДЕЛЕНИЕ С ПОМОЩЬЮ КОСИНУСНОГО ПРЕОБРАЗОВАНИЯ БОКСА-МЮЛЛЕРА

ЗАМЕЧАНИЕ: если Вы хотите напечатать подробно сформированную выборку, вводите количество пар не очень большим!

Количество пар псевдослучайных чисел, считанное из файла: 10000

Хотите ли Вы печатать сформированный результат преобразования и массивы равномерных процессов?

Введите Y или N латиницей : N

Вы ввели: N

Вы не захотели подробно печатать сформированные выборки

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д Л Я fi

ПРАВИЛО: Чтобы найти выборочную среднюю по первичным данным, нужно просуммировать все варианты и разделить полученный результат на объём совокупности.

Сумма вариант: 5044.83;

Объём совокупности: 10000;

Выборочное среднее для данной выборки: 0.504483;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д Л Я R

Сумма вариант: 5034.04;

Объём совокупности: 10000;

Выборочное среднее для данной выборки: 0.503404;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Г О С Р Е Д Н Е Г О Д Л Я

П Р Е О Б Р А З О В А Н И Я

Сумма вариант: -135.066;

Объём совокупности: 10000;

Выборочное среднее для данной выборки: -0.0135066;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И Д Л Я fi

ОПРЕДЕЛЕНИЕ: Выборочная дисперсия - среднее арифметическое квадратов отклонений всех вариант выборки от её средней.

ПРАВИЛО: Стандартное отклонение выборки - корень из выборочной дисперсии.

Хотите ли Вы выводить подробно все квадраты отклонений? Введите Y или N латиницей: N

Вы ввели: N

Вы не захотели подробно печатать квадраты отклонений.

Сумма квадратов отклонений: 3532.84;

Выборочная дисперсия для данной выборки: 0.353284;

Стандартное отклонение: 0.594377;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И Д Л Я R

Хотите ли Вы выводить подробно все квадраты отклонений? Введите Y или N латиницей: N

Вы ввели: N

Вы не захотели подробно печатать квадраты отклонений.

Сумма квадратов отклонений: 3491.05;

Выборочная дисперсия для данной выборки: 0.349105;

Стандартное отклонение: 0.590851;

Р А С Ч Ё Т В Ы Б О Р О Ч Н О Й Д И С П Е Р С И И Д Л Я

П Р Е О Б Р З О В А Н И Я

Хотите ли Вы выводить подробно все квадраты отклонений? Введите Y или N латиницей: N

Вы ввели: N

Вы не захотели подробно печатать квадраты отклонений.

Сумма квадратов отклонений: 9950.09;

Выборочная дисперсия для данной выборки: 0.995009;

Стандартное отклонение: 0.997502;

Р А С Ч Ё Т М Е Д И А Н Ы В А Р И А Ц И О Н Н О Г О Р Я Д А

П Р Е О Б Р А З О В А Н И Я

ОПРЕДЕЛЕНИЕ: Медиана вариационного ряда - значение, которое делит его на две равные части (по количеству вариант).

ПРАВИЛО: Массив с выборкой должен быть отсортирован.

Количество чисел выборки 10000 - чётное число. Делим её объём пополам: 5000

Берём среднее арифметическое 5000 и следующего за ним элемента;

Значение медианы: -0.0260337;

Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А Ц И О Н Н О Г О

Р Я Д А Д Л Я fi

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

n[i] -количество вариант, попавших в текущий интервал

ПРАВИЛО: Если варианта попадает на стык интервалов, то её следует относить в правый интервал.

Для всех интервалов, кроме последнего.

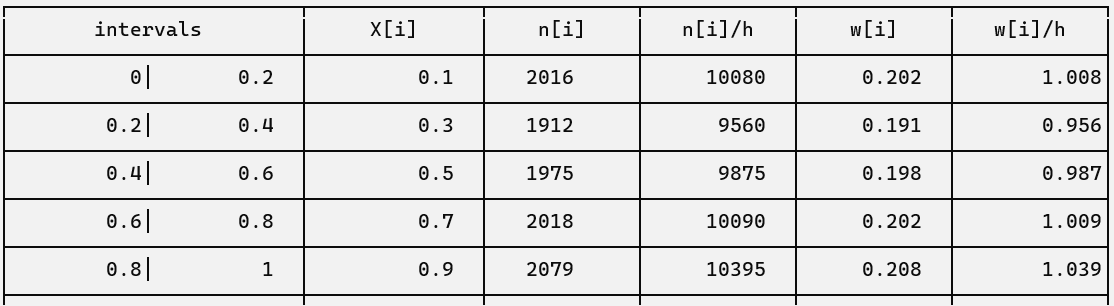
ОПРЕДЕЛЕНИЕ: Плотность частот - частота, рассчитанная на единицу ширины интервала: n[i]/h.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

САМА МАТРИЦА:



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц И И

П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я fi

Теоретические значения функции плотности равномерного распределения по формуле плотности нормального распределения

Стандартное отклонение - корень из дисперсии, sigma = 0.594377

ПРАВИЛО: Теоретические частоты вычисляются в середине каждого интервала гистограммы и добавляются

вычисления в 0 и 1. Для равномерного распределения теоретическая частота вычисляется путём деления размера выборки

на количество интервалов

ПРАВИЛО: Проверку на соответствие суммы теоретических частот общему количеству элементов осуществлять суммой

значений, вычисленных только в середине интервалов



Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я fi

ПРАВИЛО: Все значения функции, что правее конца самого левого интервала в гистограмме - имеют нулевой значение функции

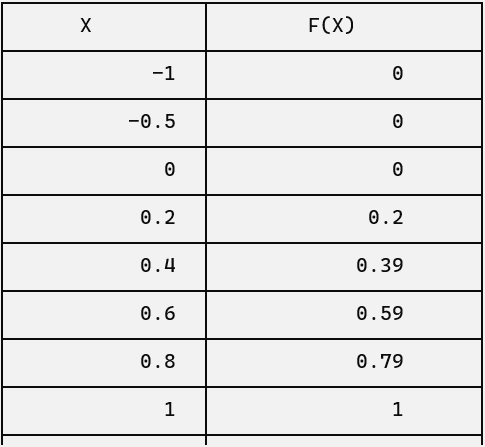
распределения далее берем крайние значения каждого интервала - за значение на оси X, накапливаем относительные частоты;

w\_n = w[i] + w\_n - значение на оси Y

Построение эмперической функции распределения

ПРАВИЛО: Значения формируются вида (x[прав]; w[i]), где w[i] -относительная частота, которая успела накопиться

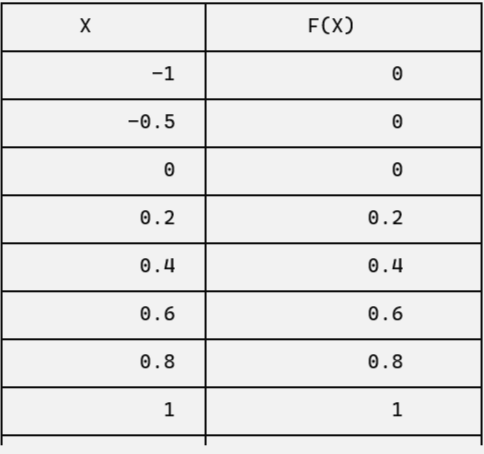
на всех пройденных интервалах



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я fi

ЗАМЕЧАНИЕ: расчёт происходит по формуле функции распределения равномерного распределения в тех же точках, что и

практическое;



Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А Ц И О Н Н О Г О

Р Я Д А Д Л Я R

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

n[i] -количество вариант, попавших в текущий интервал

ПРАВИЛО: Если варианта попадает на стык интервалов, то её следует относить в правый интервал.

Для всех интервалов, кроме последнего.

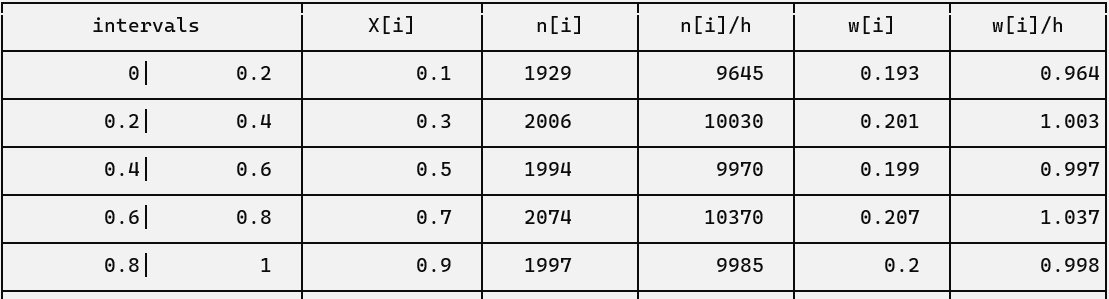
ОПРЕДЕЛЕНИЕ: Плотность частот - частота, рассчитанная на единицу ширины интервала: n[i]/h.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

САМА МАТРИЦА:



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц И И

П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я R

Теоретические значения функции плотности равномерного распределения по формуле плотности нормального распределения

Стандартное отклонение - корень из дисперсии, sigma = 0.590851

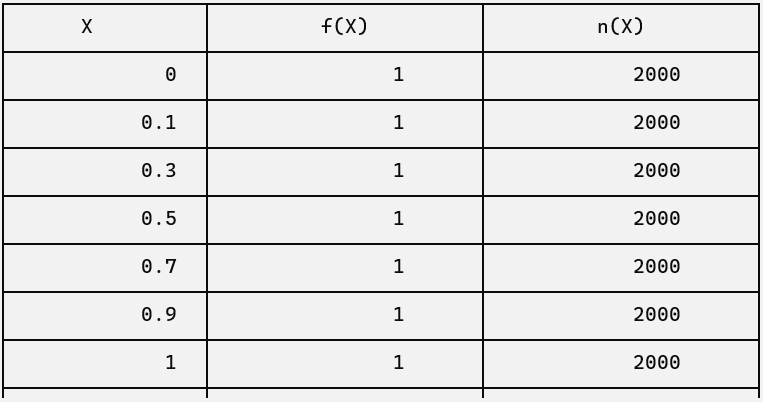
ПРАВИЛО: Теоретические частоты вычисляются в середине каждого интервала гистограммы и добавляются

вычисления в 0 и 1. Для равномерного распределения теоретическая частота вычисляется путём деления размера выборки

на количество интервалов

ПРАВИЛО: Проверку на соответствие суммы теоретических частот общему количеству элементов осуществлять суммой

значений, вычисленных только в середине интервалов



Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я R

ПРАВИЛО: Все значения функции, что правее конца самого левого интервала в гистограмме - имеют нулевой значение функции

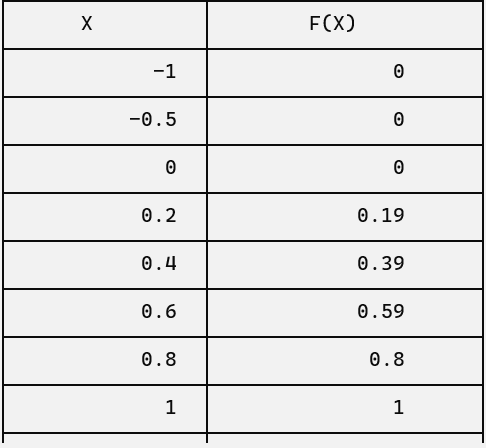
распределения алее берем крайние значения каждого интервала - за значение на оси X, накапливаем относительные частоты;

w\_n = w[i] + w\_n - значение на оси Y

Построение эмперической функции распределения

ПРАВИЛО: Значения формируются вида (x[прав]; w[i]), где w[i] -относительная частота, которая успела накопиться

на всех пройденных интервалах



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я R

ЗАМЕЧАНИЕ: расчёт происходит по формуле функции распределения равномерного распределения в тех же точках, что и

практическое;



Р А С Ч Ё Т И Н Т Е Р В А Л Ь Н О Г О В А Р И А Ц И О Н Н О Г О

Р Я Д А Д Л Я П Р Е О Б Р А З О В А Н И Я

Минимум: -3.95462

Максимум: 3.78415

X[i] - середины интервалов. Вычисляются путем вычитания из правого конца интервала левого и делением результата на 2.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

n[i] -количество вариант, попавших в текущий интервал

ПРАВИЛО: Если варианта попадает на стык интервалов, то её следует относить в правый интервал.

Для всех интервалов, кроме последнего.

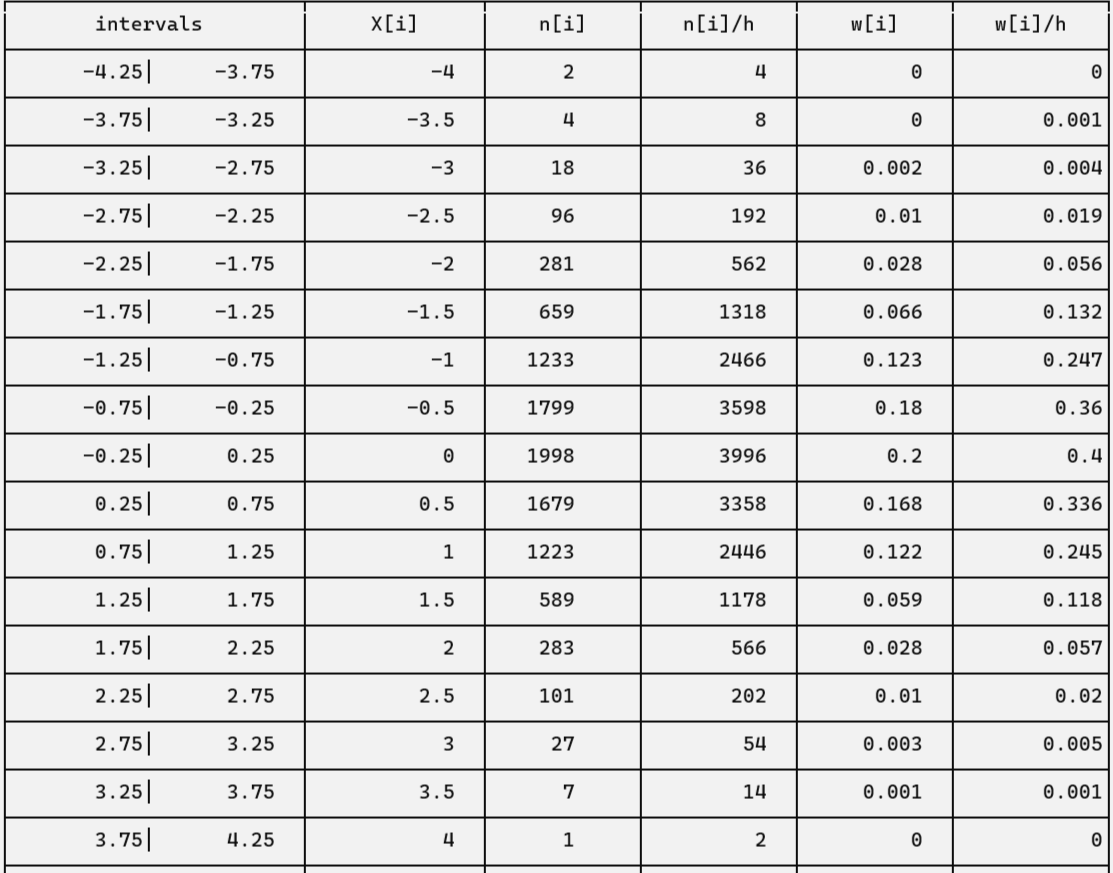
ОПРЕДЕЛЕНИЕ: Плотность частот - частота, рассчитанная на единицу ширины интервала: n[i]/h.

ОПРЕДЕЛЕНИЕ: Относительная частота реализации w[i]- отношение числа случаев, в которых встретился данный результат,

к общему числу случаев : n[i] / n.

ОПРЕДЕЛЕНИЕ: Плотность относительных частот - относительная частота, рассчитанная на единицу ширины интервала: w[i]/h.

САМА МАТРИЦА:



Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Й Ф У Н К Ц И И

П Л О Т Н О С Т И Р А С П Р Е Д Е Л Е Н И Я Д Л Я

П Р Е О Б Р А З О В А Н И Я

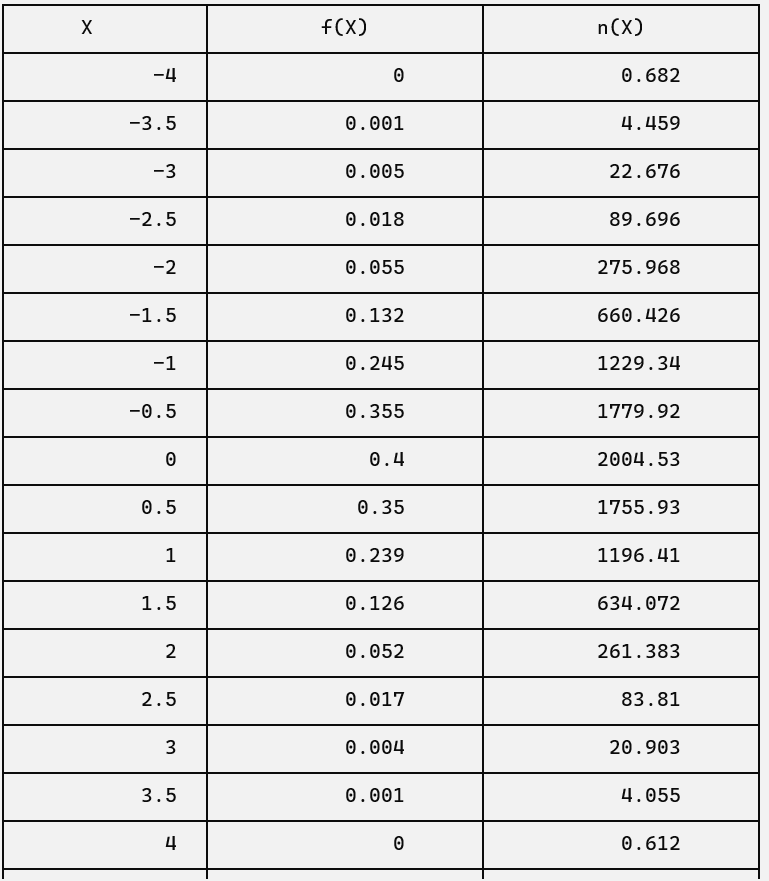
Теоретические значения функции плотности нормального распределения по вычисленному выборочному среднему и выборочной дисперсии

Стандартное отклонение - корень из дисперсии, sigma = 0.997502

ЗАМЕЧАНИЕ: Построение функции плотности распределения происходит через формулу функции плотности нормального

распределения

ПРАВИЛО: Теоретические частоты вычисляются в середине каждого интервала гистограммы



Р А С Ч Ё Т П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я

П Р Е О Б Р А З О В А Н И Я

ПРАВИЛО: Все значения функции, что правее конца самого левого интервала в гистограмме - имеют нулевой значение функции

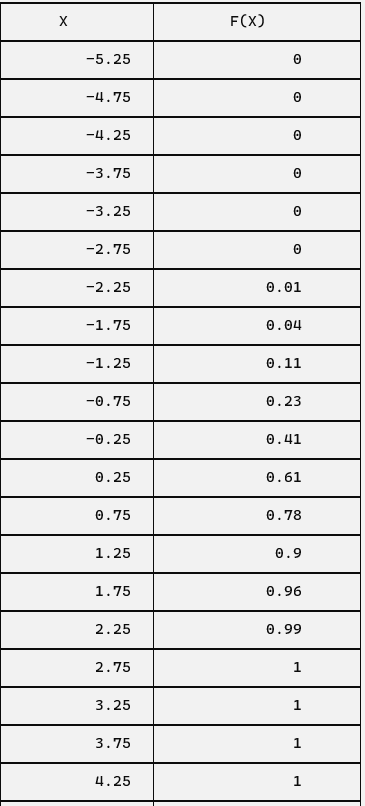
распределения алее берем крайние значения каждого интервала - за значение на оси X, накапливаем относительные частоты;

w\_n = w[i] + w\_n - значение F(X)

Построение эмперической функции распределения

ПРАВИЛО: Значения формируются вида (x[прав]; w[i]), где w[i] -относительная частота, которая успела накопиться

на всех пройденных интервалах

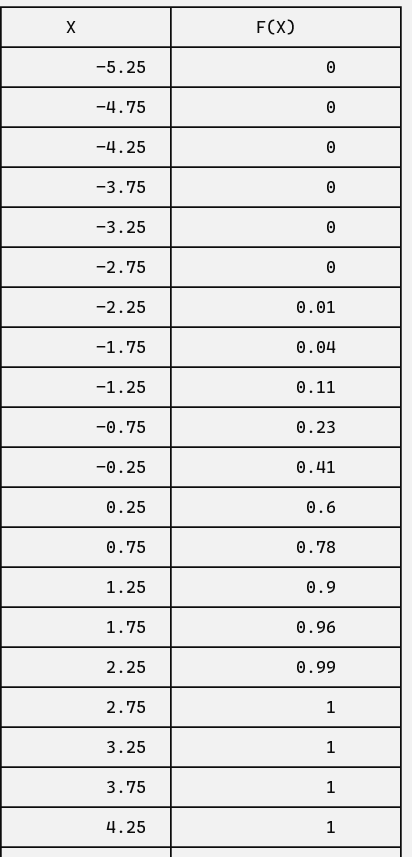


Р А С Ч Ё Т Т Е О Р Е Т И Ч Е С К О Г О П С Е В Д О С Л У Ч А Й Н О Г О П Р О Ц Е С С А Д Л Я

П Р Е О Б Р А З О В А Н И Я

ЗАМЕЧАНИЕ: расчёт происходит по формуле функции распределения гауссовского распределения в тех же точках, что и

практическое;



Р А С Ч Ё Т М О Д Ы В А Р И А Ц И О Н Н О Г О Р Я Д А

П Р Е О Б Р А З О В А Н И Я

ПРАВИЛО: Чтобы найти моду, нужно найти модальный интервал(с максимальной частотой) и воспользоваться формулой:

M = X[0]+(n[M]-n[M-1])/((n[M]-n[M-1]) + (n[M]-n[M+1]))

X[0] = -0.25 - нижняя граница модального интервала

n[M] = 1998 - частота модального интервала

n[M-1] = 1799 - частота предыдущего интервала

n[M+1] = 1679 - частота следующего интервала

МОДА: -0.0579151;

Гистограмма полученного преобразования:

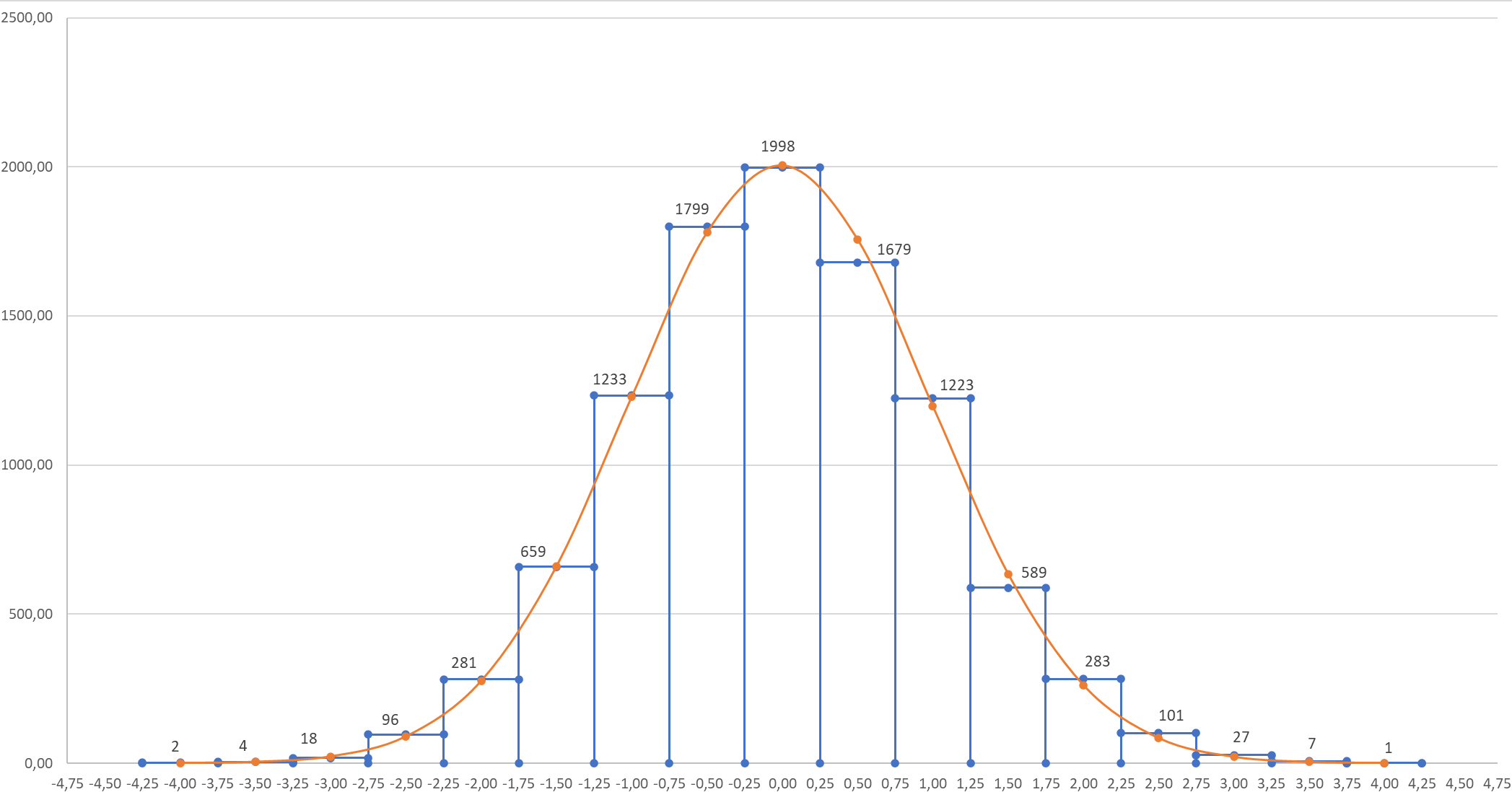
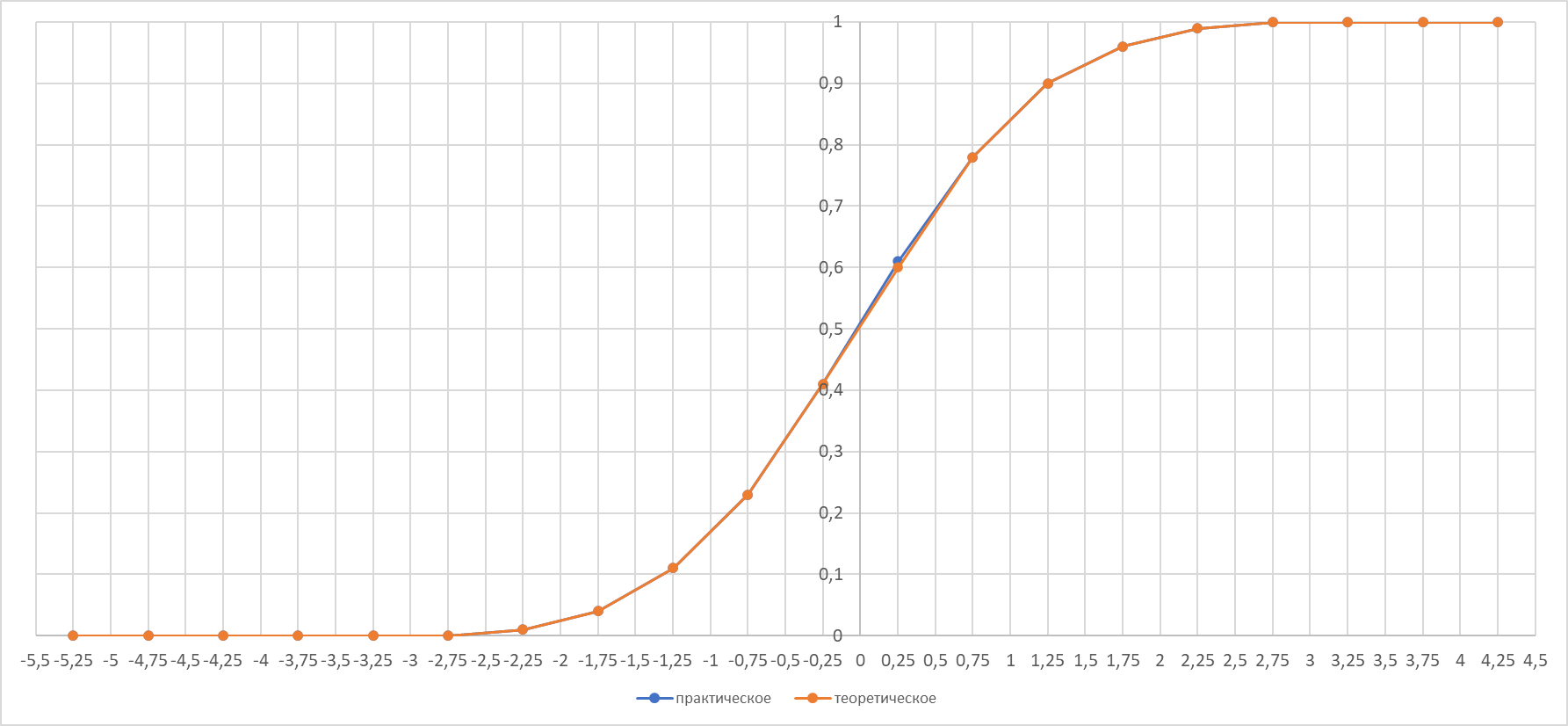


График гауссовского псевдослучайного процесса:



Гистограмма равномерно распределённой СВ fi:

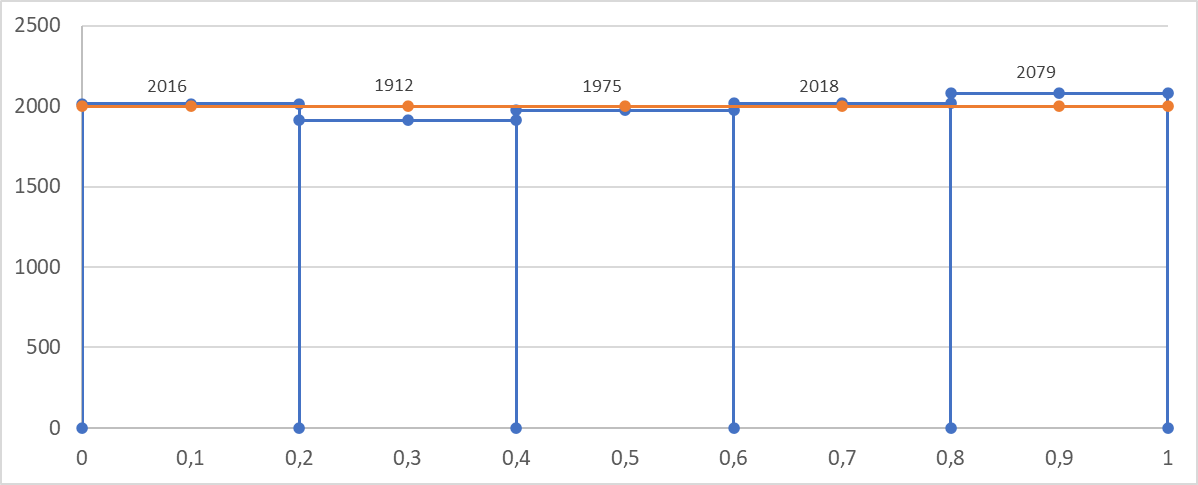
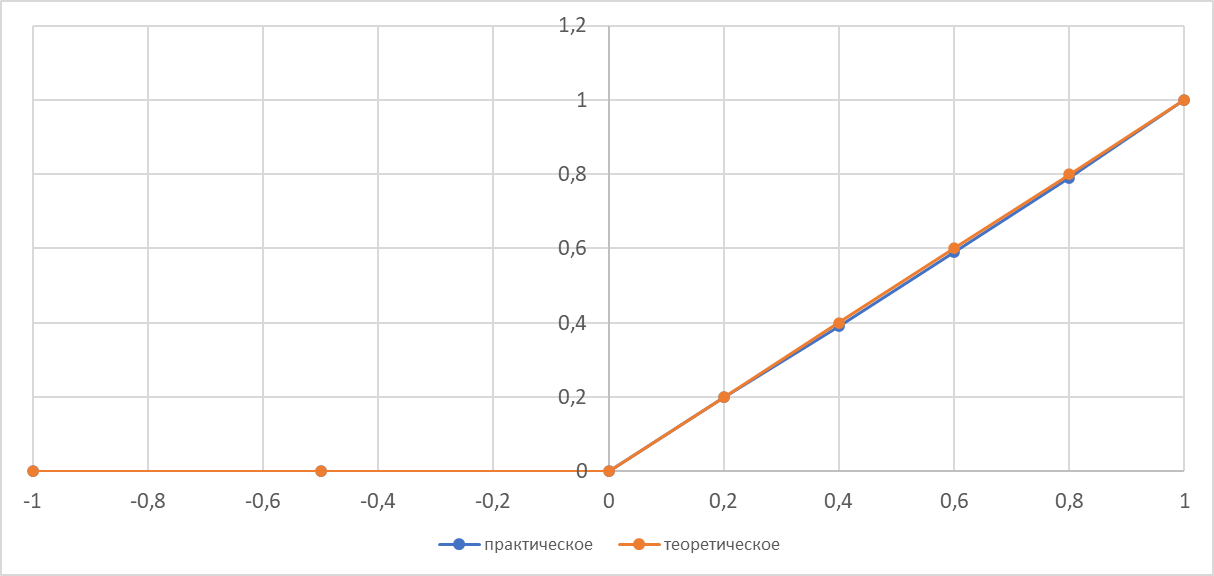


График псевдослучайного равномерного процесса СВ fi:



Гистограмма равномерно распределённой СВ R:

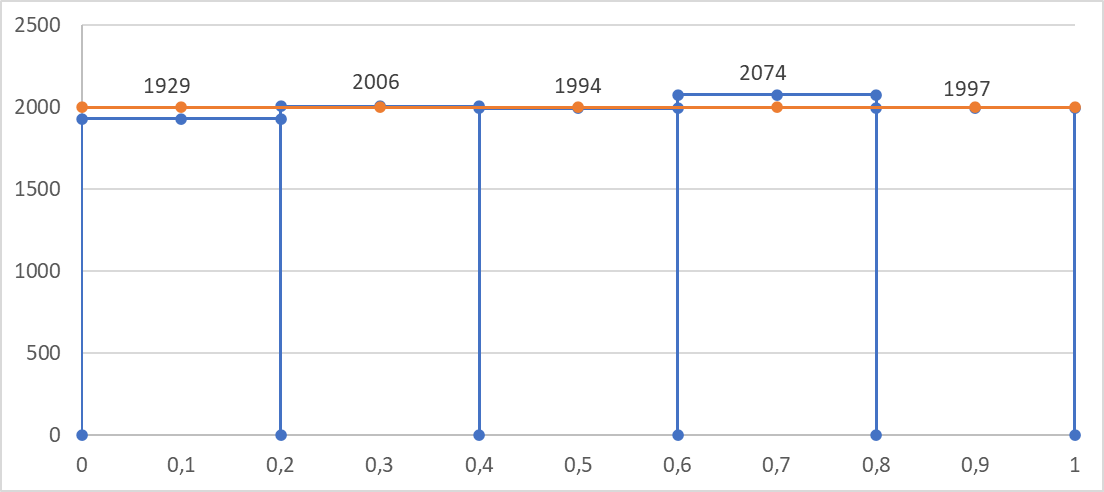
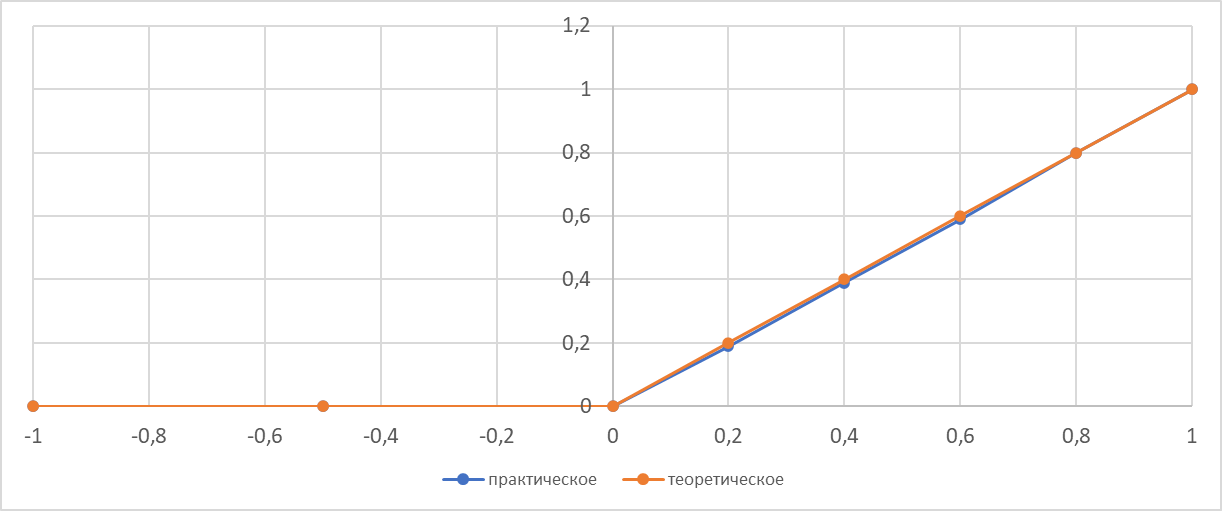


График псевдослучайного равномерного процесса СВ R:



**Объяснение вывода гистограмм и графиков на экран**

На примере выборки из 101- ой пары равномерных псевдослучайных чисел.

Принцип построения гистограммы в виде таблицы:

**Для нормального распределения**

Берём вычисленное программой значение выборочного среднего для преобразования: 0.0681991. Шаг разбиения гистограммы, установленный в программе равен 0,5.

Величины, входящие в выборку:

-3.36б, 0.654, -1.027, 1.15, -0.013, 1.182, -0.824, 1.536, -1.441, 0.986,

-0.143, 2.062, -2.211, -1.013, -0.788, -0.984, -0.756, -1.866, 0.227, -0.367,

-0.075, 0.268, 0.695, 0.114, 0.137, 0.392, 0.08, -0.774, -0.476, 0.584, 1.168, 0.828, 2.684, -0.23, -0.372, -0.088, 0.333, -0.754, 1.187, 0.597, -0.76,

-0.608, -0.68, 0.302, 0.427, 1.146, -0.427, -1.037, 1.097, -0.845, 1.859, 1.636,

-0.28, 1.18, -0.106, 1.389, -0.545, 0.796, -0.628, -1.928, -0.655, -1.45, 0.032,

-0.798, -1.381, 0.587, -0.597, 1.033, 0.284, -0.621, -0.867, -0.235,-0.172,

-0.055, -0.709, 0.128, -0.474, 0.907, 2.679, 1.003, 0.387, -0.409, 0.295, 0.419,

-0.091, 1.419, -0.817, -0.21, -0.157, -0.018, 1.021, 3.32, 0.394, 1.319, 1.084,

-2, -0.295, 0.987, 0.383, -0.214, 0.142

Далее массив выборки сортируется. Наименьшее значение в массиве: -3.362. Наибольшее значение в массиве: 3.32;

При разбитии выборки по интервалам отталкиваемся от округлённого до целого значения выборочного среднего. В нашем случае – от 0. Далее устанавливаем центральный интервал гистограммы: отступаем от нуля вправо и влево на 0,25. Центральный интервал в гистограмме имеет значения границ [-0,25;0,25).

Далее, начинаем идти вправо от левой границы центрального интервала путём вычитания шага 0,5 до момента, пока «не перескочим» наименьшее значение массива выборки:

-0,25-0,5 = -0,75;

-0,75-0,5 = -1,25;

-1,25-0,5 = -1,75;

-1,75-0,5 = -2,25;

-2,25-0,5 = -2,75;

-2,75-0,5 = -3,25;

-3,25-0,5 = -3,75 – перескочили. Тогда устанавливаем это значение за крайнее левое значение самого левого интервала. Начинаем «шагать» в положительную сторону с шагом 0,5: формируем интервалы [-3,75; -3,25). Смотрим, сколько значений из массива попало в этот интервал: 1;

Следующий интервал: [-3,25;-2,75). Значения были получены путем прибавления 0,5 к каждому концу интервала. В данный интервал попало 0 значений массива. Следующий интервал: [-2,75;-2,25) – 0 значений массива. Далее будем иметь:

[-2,25; -1,75) = 4 элемента массива

[-1,75; -1,25) = 3 элемента массива

[-1,25; -0,75) = 14 элементов массива

[-0,75; -0,25) = 16 элементов массива

[-0,25; 0,25) = 21 элемент массива

Затем в программе вычисляется значение крайнего правого значения интервала путем прибавления 0,5 к текущему значению до момента, пока не «не перескочим» больший элемент массива выборки:

0,25+0,5 = 0,75;

0,75+0,5 = 1,25;

1,25+0,5 = 1,75;

1,75+0,5 = 2,25;

2,25+0,5 = 2,75;

2,75+0,5 = 3,25;

3,25+0,5 = 3,75; - перескочили. Тогда устанавливаем это значение за крайнее правое значение самого правого интервала.

Аналогично проходим все интервалы с шагом 0,5 до момента, когда крайнее правое значение очередного интервала не станет равным 3,75. Смотрим, сколько значений выборки попало в тот или иной интервал:

[0,25; 0,75) = 16 элементов массива

[0,75; 1,25) = 16 элементов массива

[1,25; 1,75) = 5 элементов массива

[1,75; 2,25) = 2 элемента массива

[2,25; 2,75) = 2 элемента массива

[2,75; 3,25) = 0 элементов массива

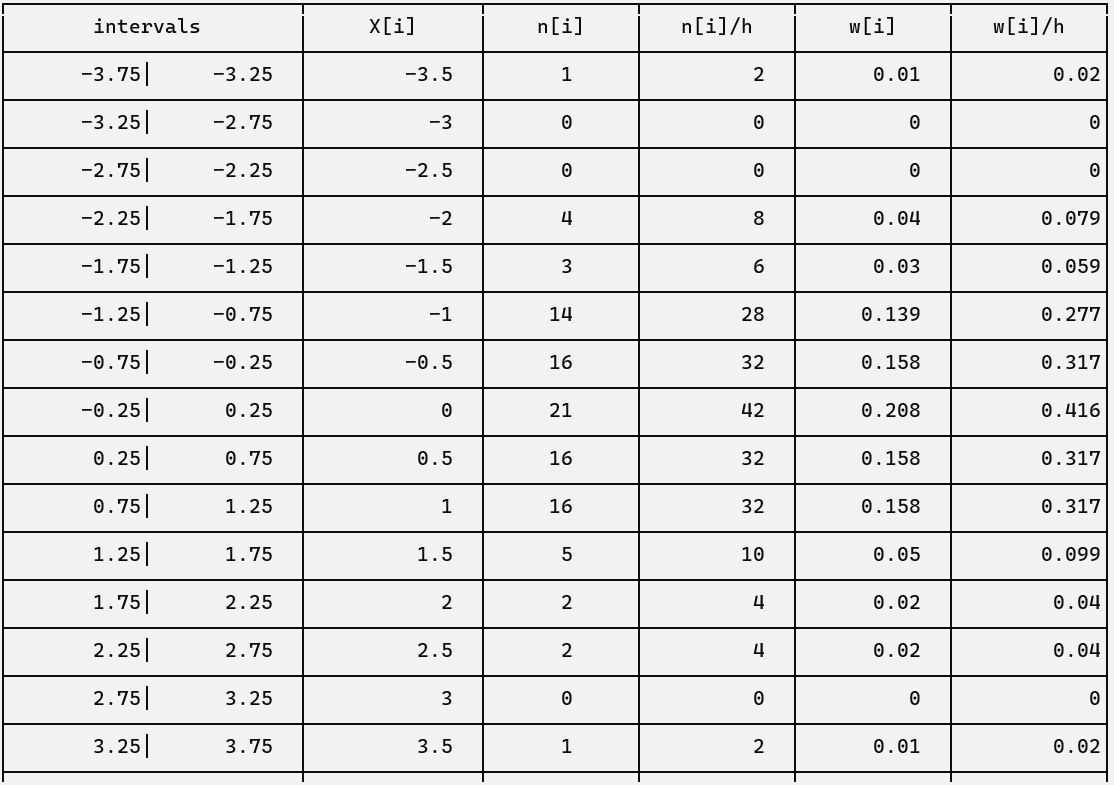
[3,25; 3,75] = 1 элемент массива

Полученные количества элементов массива в каждом интервале являются частотами для данного интервала. Далее программа находит плотность частот для каждого интервала путём деления частоты на шаг, середины каждого интервала, относительные частоты и их плотность.

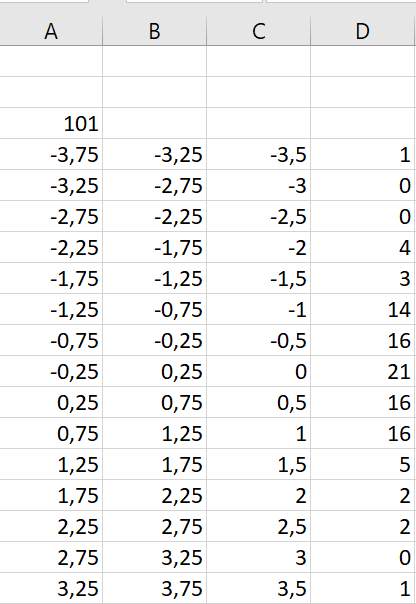
Все вышеперечисленные данные программа выводит в виде таблицы которая имеет такой же вид, как и матрица для гистограммы на экране в файл,.

Далее открываем программу Excel, куда заносим начало интервала в первый столбец, конец интервала – во второй столбец, середину интервала – в третий столбец, частоту – в четвертый столбец.

В итоге данная таблица переходит

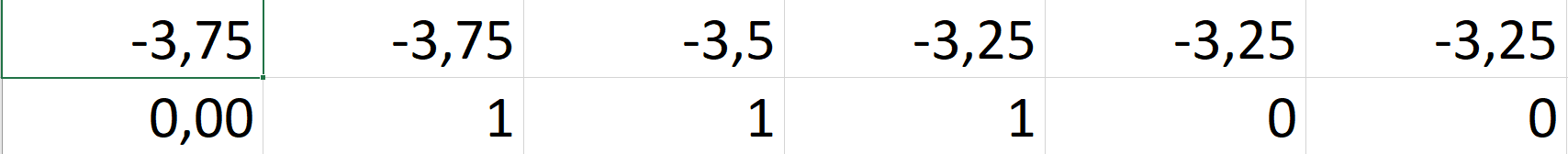


Данная таблица переходит в такую в программе:

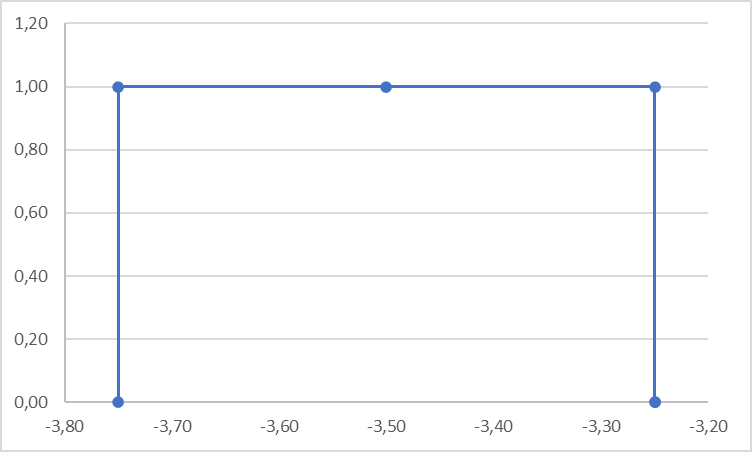


Затем начинаем строить поточечно гистограмму. Для этого выписываем в отдельную ячейку в программе значение -3,75. Это – начало гистограммы. Под ним выписываем 0. Далее в соседней ячейке выписываем это же значение и ставим под ним значение частоты для данного интервала: 1 – поднимаемся на нужную высоту интервала. Оно сохраняется и в середине интервала, потому в соседнюю ячейку помещаем значение -3, 5. Под ним: 1. Аналогично поступаем и с -3,25. Т.о. мы просмотрели интервал целиком. Нам необходимо его «замкнуть», поэтому дублируем значение -3,25 в соседней ячейке и под ним ставим 0. Но, т.к крайнее правое значение интервала фигурирует и в следующем интервале, то еще раз дублируем значение -3,25, но под ним уже пишем частоту следующего интервала : 0. Затем выделяем все заполненные ячейки и выбираем поле «Вставка» - «Вставить точечную диаграмму» - «Точечная с прямыми отрезками и маркерами».

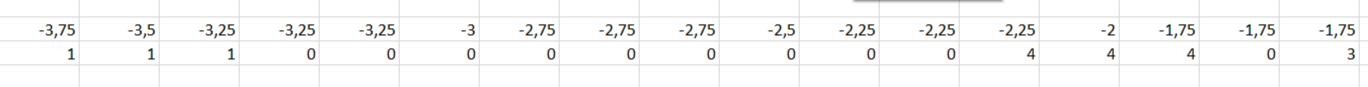
Все проделанные действия в программе имеют следующий вид:

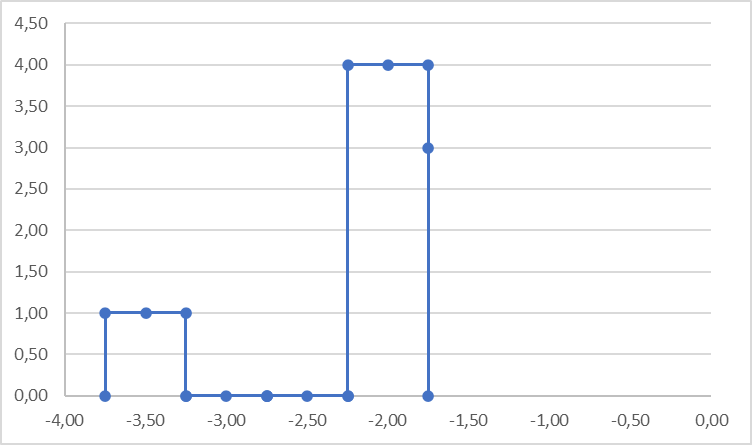


Мы построили один полный столбец гистограммы и начало другого:

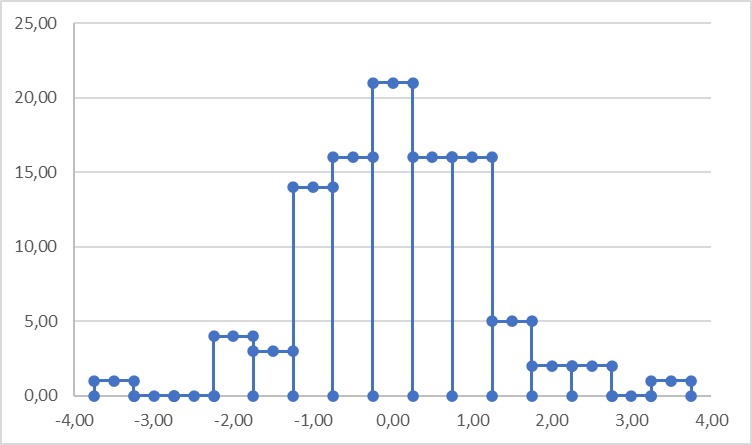


Получили самый первый интервал гистограммы. Аналогичную операцию проделываем и с оставшимися интервалами. Так, после добавления 4-го интервала гистограмма примет вид:

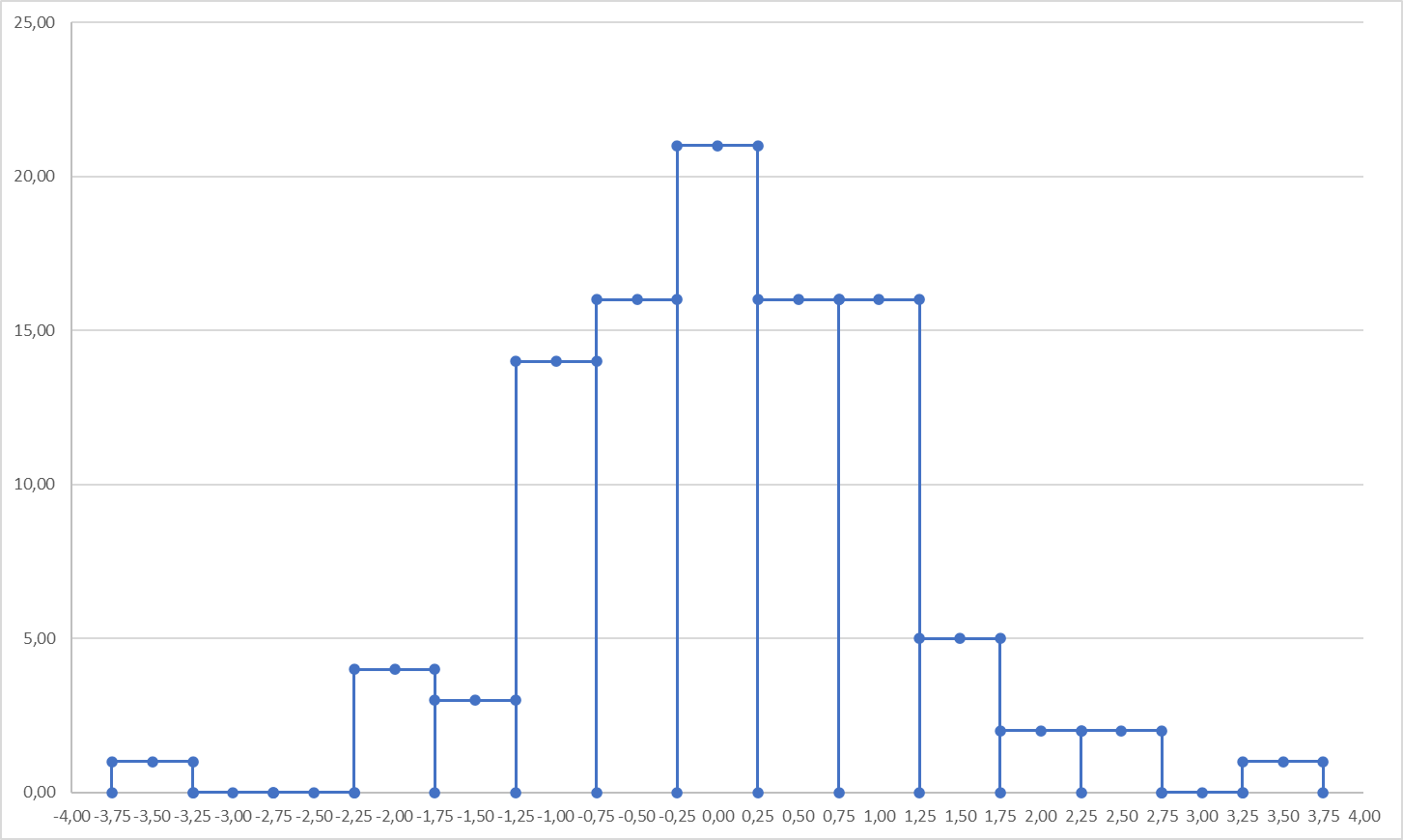




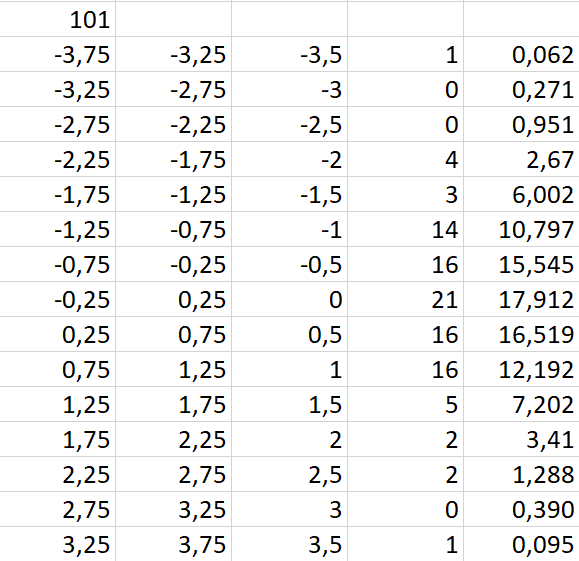
А после добавления всех интервалов гистограмма примет вид:



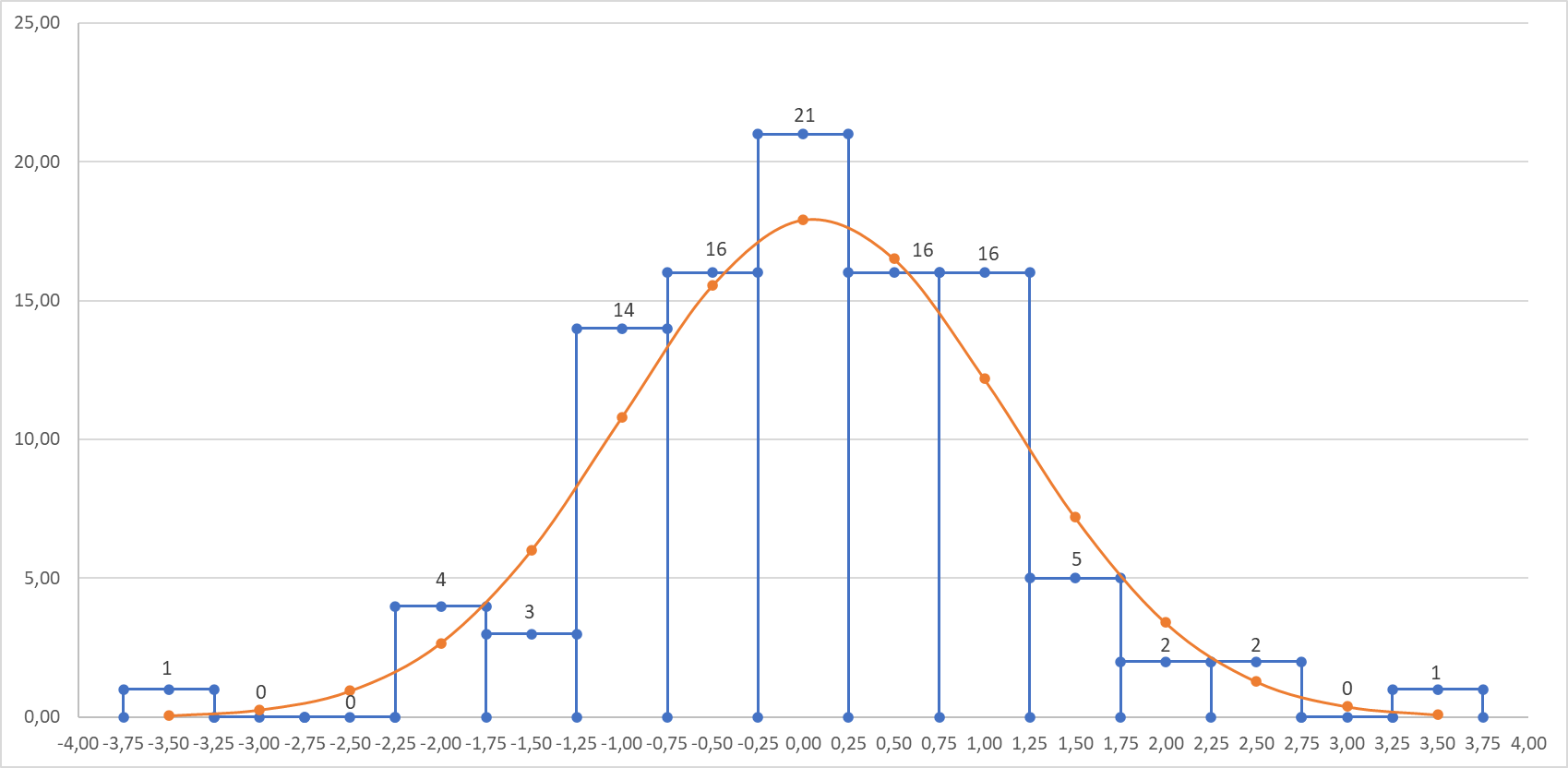
Далее нажимаем на ось Ox- выбираем цену деления, равную половине шага разбиения, т.е 0,25.



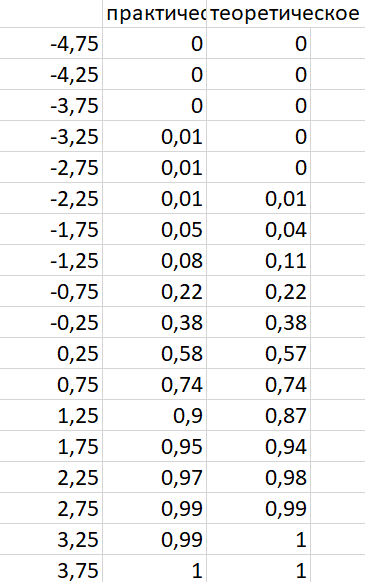
Из выходного файла копируем значения n из пункта «расчёт теоретической функции плотности распределения для преобразования». В этом разделе происходит расчёт функции плотности распределения по значению x, равному середине интервала и теоретических частот n по функции плотности. Помещаем эти значения в 5 -ый столбец таблицы:



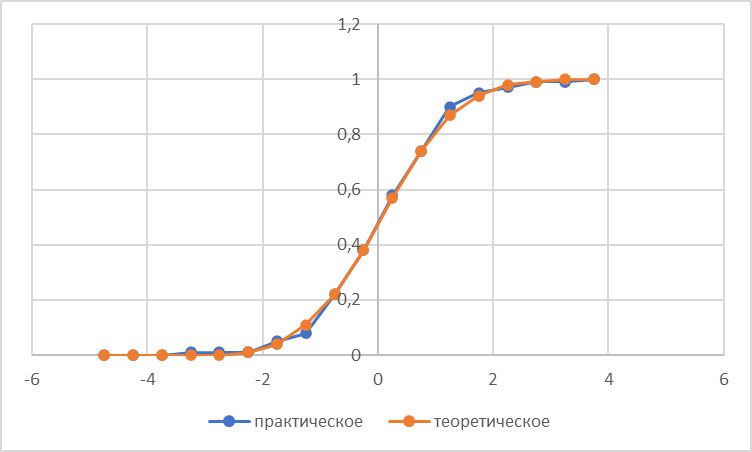
Далее нажимаем правой кнопкой мыши на уже построенную гистограмму частот и выбираем «Выбрать данные» - «Добавить». В открывшемся окне в поле «Значения X» выделяем все середины интервалов в таблице. В поле «Значения Y» выделяем весь 5-ый столбец таблицы. Нажимаем «ок». В результате появляется график в виде ломанных, поэтому необходимо его сгладить В появившемся графике поверх гистограммы, нажав правую кнопку мыши, выбираем «Изменить тип диаграммы» - «Точечная с гладкими кривыми и маркерами». В итоге, получается такая гистограмма:



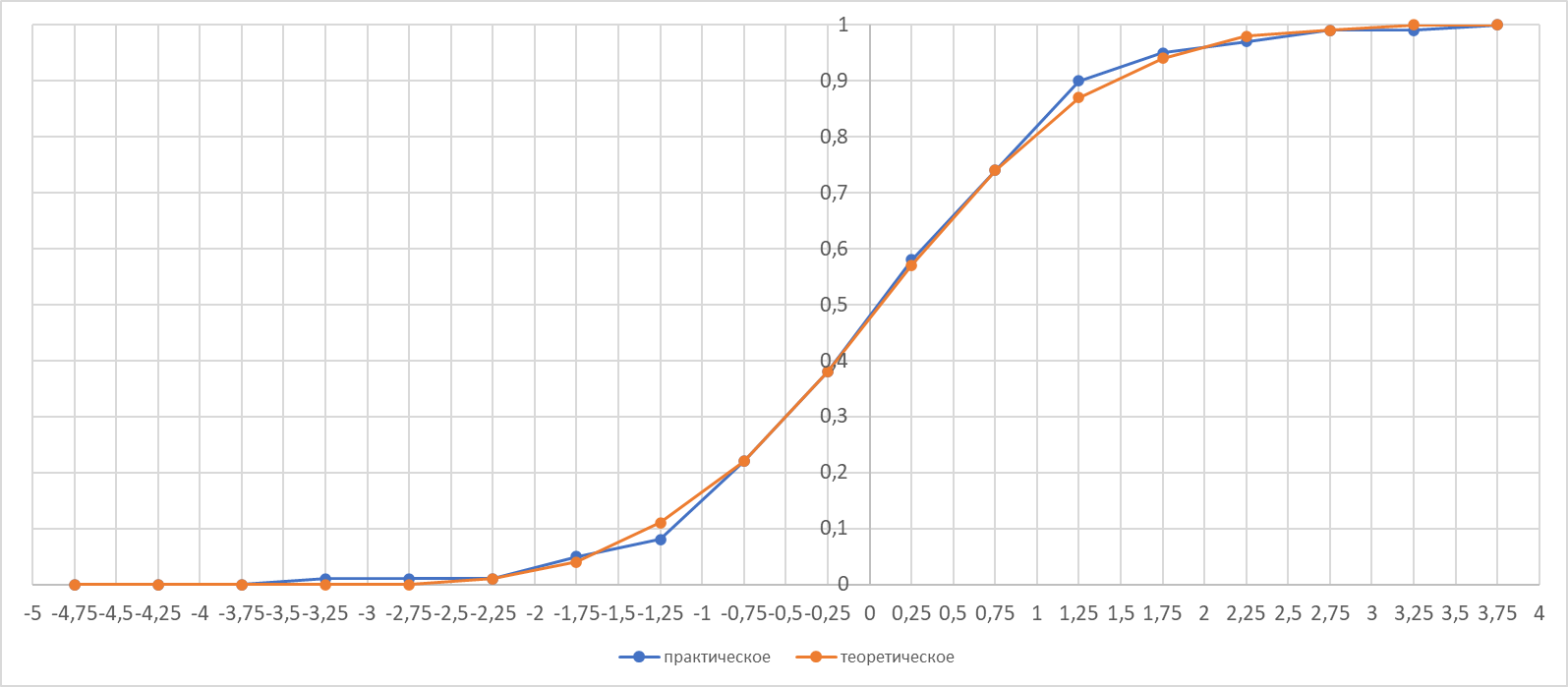
Далее необходимо построить теоретический и практический графики гауссовского процесса. Для этого из выходного файла копируем значения из первой колонки таблички в пункте «Расчёт псевдослучайного процесса для преобразования» в один столбик программы. Значения практической функции – во второй столбец. Так как функции теоретического и практического процессов вычисляются в одинаковых точках, то в 3-ий столбец помещаем теоретические значения функции из пункта «Расчёт теоретического псевдослучайного процесса для преобразования». В итоге, получаем:



Далее выделяем всю таблицу вместе с надписями, нажимаем «Вставка» - «Вставить точечную диаграмму» - «Точечная с прямыми отрезками и маркерами». В результате, выводится:



Нажимая на ось X, в выпавшем меню выставляем цену деления, равной половине шага интервала, т.е. 0, 25. Нажимая на ось Y, в выпавшем меню выставляем наибольшее значение 1. В результате, график примет вид:



Аналогично строятся остальные гистограммы и графики как для гауссовских, так и для равномерных распределений.

**Вывод**

Реализована программа, которая на основании косинусного преобразования Бокса-Мюллера моделирует гауссовское распределение с помощью встроенного генератора псевдослучайных чисел.

Для написания кода программы была использована специальная литература, включающая в себя учебники и статьи по программированию. В курсовой работе использованы изученные мной конструкции С++, а именно:

а) для написания функций:

-прототипы функций и предварительное объявление

-передача аргументов по значению, указателю, ссылке

б) для выполнения арифметических операций:

-арифметические операторы

-стандартная библиотека математических операторов cmath.

в) для порядка выполнения кода:

-операторы условного ветвления

-цикл for

-оператор exit(1)

-оператор return

г) для работы с данными:

-встроенные типы данных

-глобальные константы PI, FNAME.

-указатели, ссылки

-логические операторы: ИЛИ.

д) для ввода и вывода данных:

-поточный ввод и вывод данных с помощью функций библиотек iostream, fstream.

-использование символов таблицы ASCII с помощью функций библиотеки iomanip.

-использование подкрашивания текста при помощи управляющих последовательностей ANSI.

-использование управляющих символов: \n;\t

-использование оператора round библиотеки cmath

е) для работы с динамическими данными

-указатели

-адресная арифметика указателей

- операции new, delete

- матрица указателей

ж) для работы с псевдослучайными числами

- операция rand()

- значение RAND\_MAX

з) для работы с массивами:

- указатели для передачи в функции

- циклы for прохода по массиву

- алгоритмы сортировки(сортировка слиянием)

- адресная арифметика

- использование заведомо больших целых чисел : INT\_MAX.

и) для работы с алгоритмом Бокса-Мюллера:

- готовая формула косинусного преобразования

- создание двух псевдослучайных чисел.

к) для работы над созданием гистограммы:

- модифицированная формула квадратного расчёта максимального

количества интервалов

л) для работы с теоретической функцией плотности распределения:

- формула плотности нормального и равномерного распределений;

- библиотека cmath;

Как можно видеть из полученных гистограмм, качество как равномерных распределений, так полученных нормальных распределений растёт с увеличением размера выборки (гистограмма все больше начинает приобретать форму построенной теоретической плотности распределения, меньше становится разность между наибольшим и наименьшим столбцом гистограммы по сравнению с размером выборки). Сами функции распределения псевдослучайных процессов с ростом выборки все меньше отличаются от теоретических функций распределений, что говорит о том, что качество выборки растёт по мере приближения к генеральной совокупности. Причём качество полученных равномерных распределений не является идеальным, а является довольно приемлемым (можно даже сказать хорошим), чтобы рассматривать значения как равномерно распределённые. Качество не идеально (количество элементов в каждом интервале гистограммы не совпадает с теоретическим значением) из-за того, что в программе используется встроенный равномерный «псевдогенератор», который генерирует далеко не равное количество значений для каждого интервала. Если бы мы имели не выборку, а генеральную совокупность всех равномерно распределённых величин, то все псевдослучайные процессы были бы идеальными.

**Список источников**

1. А.И.Кибзун, Е.Р.Горяинова, А.В.Наумов, А.Н.Сиротин.ТЕОРИЯ ВЕРОЯТНОСТЕЙ И МАТЕМАТИЧЕСКАЯ СТАТИСТИКА. БАЗОВЫЙ КУРС С ПРИМЕРАМИ И ЗАДАЧАМИ. - М.: ФИЗМАТЛИТ, 2002. - 224 с.
2. Ивченко Г.И., Медведев Ю.И. Математическая статистика. - М.: Высшая школа, 1984.
3. Джордж Е. П. Бокс, Мервин Э. Мюллер, «Заметка о генерации случайных нормальных отклонений», The Annals of Mathematical Statistics Vol. 29, No. 2 (июнь 1958 г.), стр. 610-611.
4. Асриев А. В., Кибзун А.И. Практикум по статистическому моделированию на ЭВМ. – М.: Изд-во МАИ, 1989.
5. Преобразование равномерно распределенной случайной величины в нормально распределенную: [Электронный ресурс] // habr.com., 2006-2022. URL: <https://habr.com/ru/post/208684/> (Дата обращения: 14.11.2022).

# 6. Стивен Прата, «Язык программирования C++. Лекции и упражнения» - М. . . : 000 "И.Д. Вильямс", 2018.

## 7. Чтение и запись текстовых файлов: [Электронный ресурс] // metanit.com.,

2012-2022. URL: https://metanit.com/cpp/tutorial/8.3.php (Дата обращения: / / / / 16.11.2022).

8. [Сандлер Кори](https://www.labirint.ru/authors/73369/), [Баджетт Том](https://www.labirint.ru/authors/166490/), [Майерс Гленфорд](https://www.labirint.ru/authors/166489/), «Искусство тестирования / программ». – М.: Диалектика, 2020.

9. Интервальный вариационный ряд. Гистограмма относительных частот: / / / / / [Электронный ресурс]//mathprofi.ru., 2010-2022. URL: //////// //////http://mathprofi.ru/intervalnyi\_variacionnyi\_ryad.html (Дата обращения:

20.11.2022).

# 