Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Тульский государственный университет»

КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

**ГЕНЕРАЦИЯ НОРМАЛЬНОГО И ЗАДАННОГО РАСПРЕДЕЛЕНИЙ**

отчет о   
лабораторной работе №2

по дисциплине

*ТЕХНОЛОГИИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ*

***ВАРИАНТ 6***

|  |  |  |
| --- | --- | --- |
| Выполнила: | ст. гр. 230711 | Павлова В.С. |
| Проверил: | асс. каф. ИБ | Курбаков М.Ю. |

Тула, 2022 г.

# **ЦЕЛЬ И ЗАДАЧА РАБОТЫ**

**Цель:** изучить генерацию случайных величин по заданному и нормальному законам распределения.

**Задача:** в данной работе требуется написать программы, демонстрирующие использование изученных принципов.

# **ЗАДАНИЕ НА РАБОТУ**

1. С помощью метода обратной функции получить случайную величину с заданной по варианту плотностью распределения *f*(*x*), график которой приведён на рисунке 1.
2. Построить нормальное распределение с заданными математическим ожиданием и дисперсией. Полученную в результате генерирования плотность вероятности сравнить с теоретической.

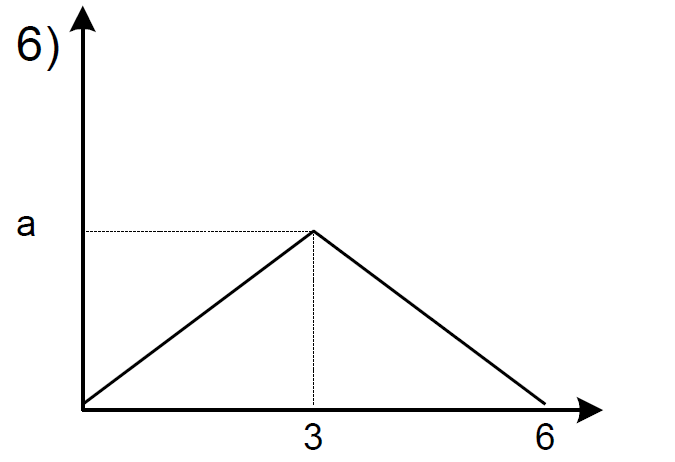


Рисунок 1 – График функции плотности распределения

# **СХЕМА ПРОГРАММЫ**

1. Схема алгоритма, предназначенного для генерации заданного распределения, представлена на рисунке 2.



Рисунок 2 – Схема алгоритма для генерации заданного распределения

1. Схема алгоритма, предназначенного для генерации нормального распределения, представлена на рисунке 3.



Рисунок 3 – Схема алгоритма для генерации нормального распределения

# **ТЕКСТ ПРОГРАММЫ**

# Текст программы на языке программирования С++ для генерации заданного распределения представлен в листинге 1.

## **Листинг 1. Текст программы**

#include <iostream>

#include <cmath>

#include <map>

double Rnd(int\* x0) //(0;1)

{

int c = 15, m = 65536, a = 13;

int val = (a \* (\*x0) + c) % m;

\*x0 = val;

return (float)val / m;

}

int main()

{

setlocale(LC\_ALL, "Russian");

int n, x0 = 12;

int\* adressX0 = &x0;

float x, y, sumY = 0;

std::map <float, int> crd;

std::cout << "\t\tВведите длину заданного распределения: ";

std::cin >> n;

std::cout << "\n";

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

{

y = Rnd(adressX0);

if (y <= 0.5) x = 3 \* sqrt(2 \* y);

else x = 6 - 3 \* sqrt(2 - 2 \* y);

crd[floor(x \* 10) / 10]++;

} std::cout << "\n";

for (auto to : crd)

std::cout << to.first << "\t

<< (float)to.second / n << "\n";

return 0;

}

Текст программы на языке программирования С++ для генерации нормального распределения представлен в листинге 2.

## **Листинг 2. Текст программы**

#include <iostream>

#include <fstream>

#include <cmath>

## **Листинг 2. Текст программы (продолжение)**

#include <map>

double Rnd(int\* x0) //(0;1)

{

int c = 15, m = 65536, a = 13;

int val = (a \* (\*x0) + c) % m;

\*x0 = val;

return (float)val / m;

}

int main()

{

setlocale(LC\_ALL, "Russian");

int n, m, x0 = 12;

float mathExp, disp;

int\* adressX0 = &x0;

float x, y, sumY = 0;

std::map <float, int> crd;

std::cout << "\t\tВведите длину нормального распределения: ";

std::cin >> n;

std::cout << "\t\tВведите дисперсию и математическое ожидание: ";

std::cin >> mathExp >> disp;

float RMSD = sqrt(disp);

std::cout << "\n";

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

{

sumY = 0;

for (size\_t i = 0; i < 12; i++)

{

sumY += Rnd(adressX0);

}

x = RMSD \* (sumY - 6) + mathExp;

crd[floor(x \* 10) / 10]++;

} std::cout << "\n";

std::map<float, int>::iterator to = crd.begin();

for (size\_t i = 0; i < crd.size(); i++)

{

std::cout << to->first << "\t" << (float)to->second / n << "\n";

to++;

} std::cout << "\n\n";

return 0;

}

# **ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ**

Первая программа предназначена для генерации заданного распределения. При запуске программы пользователю предлагается ввести длину последовательности. После программа формирует последовательность чисел заданной длины, а также рассчитывает координаты точек для построения графика плотности вероятности и выводит эти координаты на экран.

Вторая программа работает аналогично первой. При запуске пользователю так же предлагается ввести длину последовательности, а ещё математическое ожидание и дисперсию. После программа формирует последовательность заданной длины, а также рассчитывает координаты точек для построения графика плотности вероятности и выводит их.

# **ИНСТРУКЦИЯ ПРОГРАММИСТА**

Структуры данных, используемые в обеих программах, аналогичны, поэтому все они приведены в одной таблице (таблица 1).

Таблица 1 – Структуры данных в программе

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип (класс)** | **Предназначение** |
| n | int | Длина последовательности |
| crd | map | Структура для расчёта точек графика плотности распределения |
| x | float | Величина |
| x0 | int | Начальное число для линейного конгруэнтного генератора |
| adressX0 | \*int | Адрес ячейки памяти x0 |
| y | float | Величина, полученная линейным конгруэнтным генератором |
| sumY | float | Счётчик суммы чисел |
| mathExp | float | Математическое ожидание |
| disp | float | Дисперсия |
| RMSD | float | Среднеквадратичное отклонение (сигма) |

В обеих программах имеется следующая подпрограмма:

1. double Rnd(int \*x0) – функция, генерирующая случайное число с использованием линейного конгруэнтного метода. Структуры данных, используемые в подпрограмме, приведены в таблице 2.

Таблица 2 – Структуры данных, используемые в подпрограмме Rnd()

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Предназначение** |
| *формальные параметры* | | |
| x0 | \*int | Ссылка на предыдущее полученное число |
| *локальные переменные* | | |
| c, a, m | const int | Параметры линейного конгруэнтного генератора |
| val | double | Величина |

# **ДЕМОНСТРАЦИОННЫЙ ПРИМЕР**

1) Для получения случайной величины с заданной по варианту плотностью распределения методом обратной функции сперва проведём аналитические расчёты и определим эту обратную функцию. Согласно заданию варианта, плотность распределения имеет вид, представленный на рисунке 4:

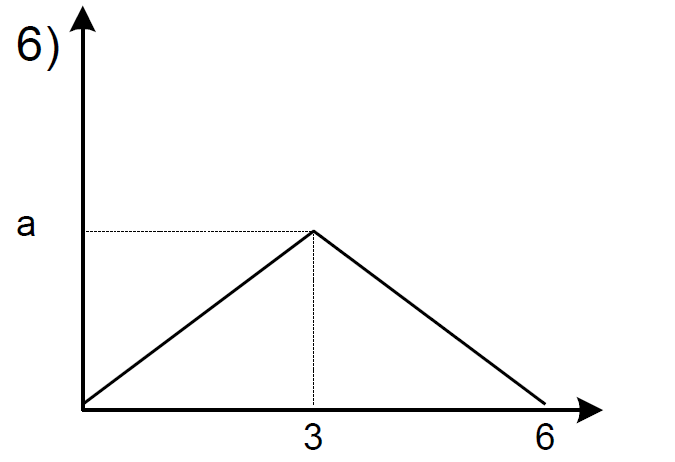


Рисунок 4 – Плотность распределения

Прежде всего, определим постоянную *а*, используя условие нормировки . Поскольку интеграл является площадью под графиком, то его можно определить как S = Здесь – сторона треугольника, равная 6. Отсюда Запишем теперь функцию распределения, учтя константу а:

Далее найдём функцию распределения *F*(*x*) = :

Теперь найдем функцию, обратную к *F*(*x*). Для этого выразим через *y*:

1. y =, отсюда

Пересчитаем область определения, подставив значения = 0, = 3 в полученное уравнение *x* = . Если = 0, тогда *y* = 0. Если же = 3, тогда имеем *y* = 0.5.

Отсюда первое уравнение из системы: = ,

1. *y* = , домножим на 18 и получим

, отсюда  
 = = 6 3,

= = 6 + 3.

Пересчитаем область определения, подставив значения = 3, = 6 в уравнение = 6 3. Если = 3, тогда *y* = 0.5. Если = 6, тогда y = 1.

Попробуем подставить = 3 во второе уравнение, которое имеет вид = 6 + 3. Это уравнение не подходит, т.к. не будет иметь решений, ведь квадратный корень не может быть отрицательным. Тогда второе уравнение системы – это уравнение = , а обратная функция имеет вид:

С помощью полученных формул программа, приведённая в листинге 1, производит расчёты. Рассмотрим теперь результат работы этой программы для n = 600000 чисел, он приведён на рисунке 5.

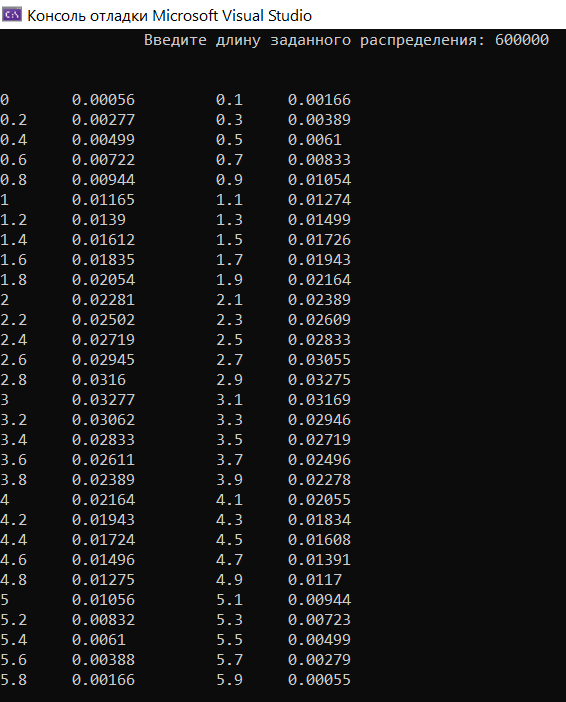
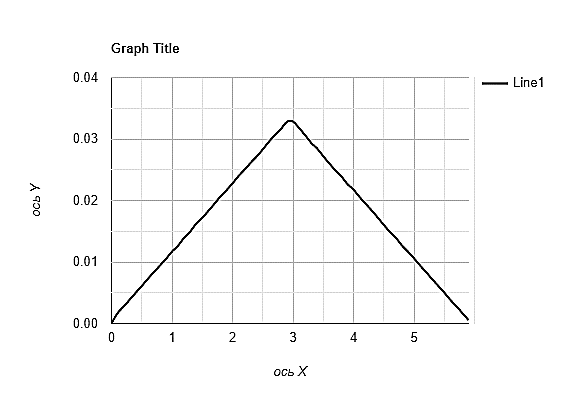


Рисунок 5 – Результат работы программы

Необходимо сравнить полученную плотность распределения с заданной по варианту. Для проверки полученного результата построим график плотности распределения, согласно значениям, полученным в программе.

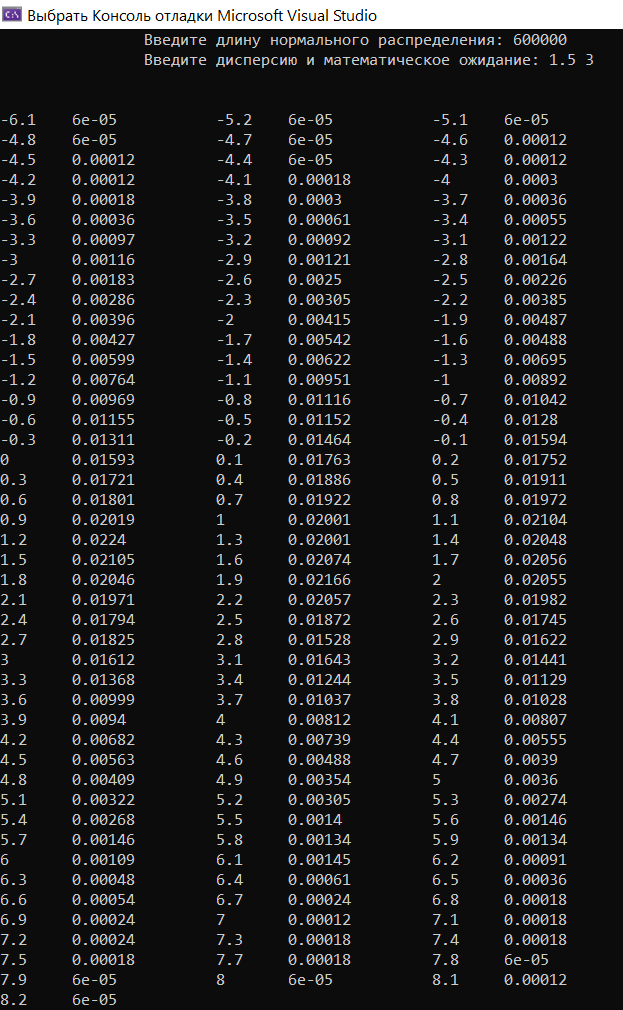
  
Рисунок 6 – График плотности заданного распределения

Промежуточный вывод: как видно по рисунку 6, есть незначительное отклонение от теоретического, однако график в целом соответствует теоретической плотности распределения.

2) Исходя из рисунка 1, приведённого в задании на работу, найдем математическое ожидание и дисперсию для заданного распределения:

.

Для генерации нормального распределения воспользуемся этими параметрами и формулой *,* где , а – независимая случайная равномерная величина.

  
  
Рисунок 7 – Результат работы программы

Для проверки полученного результата построим график для нормального распределения, согласно значениям, полученным в программе. Этот график приведён на рисунке 8.

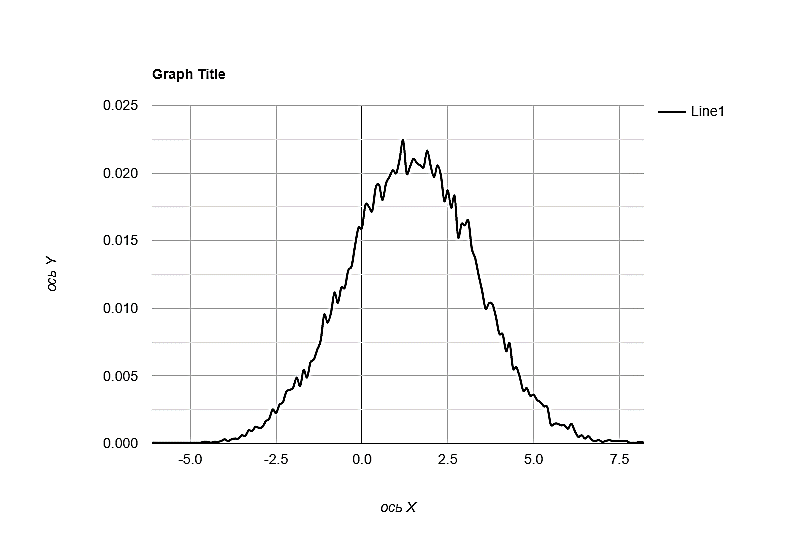


Рисунок 8 – Графики плотности полученного распределения

Теперь необходимо построить кривую нормального распределения, рассчитав значения функции для заданных параметров математического ожидания и дисперсии. Полученный график приведен на рисунке 9.

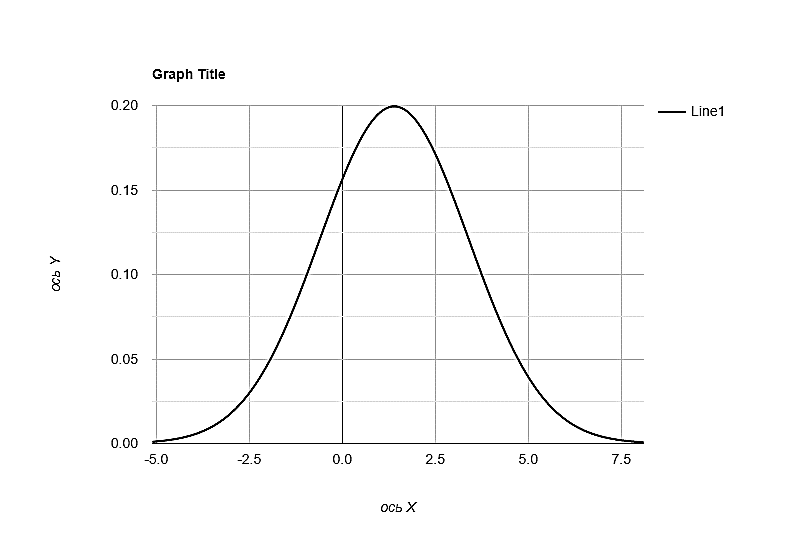


Рисунок 9 – Кривая распределения Гаусса для заданных параметров математического ожидания и дисперсии

Промежуточный вывод: как видно по рисункам 8 и 9, общий вид функции плотности полученного нормального распределения соответствует кривой распределения Гаусса с некоторым отклонением.

# **ВЫВОДЫ**

В ходе данной лабораторной работы был изучен принцип генерации случайных величин по заданному и нормальному законам распределения. Для демонстрации полученных знаний была написаны программы для генерации соответствующих распределений, результат работы которой был проверен аналитически. В ходе проверки с помощью плотностей распределения обнаружилось, что полученные распределения соответствуют теоретическим с незначительным отклонением.