Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по Лабораторной работе №1

по курсу «Моделирование»

на тему: «Исследование функций и плотностей распределения случайных величин»

Студент ИУ7-72Б (Группа) ________ (Подпись, дата) Т. М. Сучкова (И. О. Фамилия)

Преподаватель ________ (Подпись, дата) И. В. Рудаков (И. О. Фамилия)

2022 г.

1 Задание

Разработать программу для построения графиков следующих функций и плотностей распределения случайных величин.

1. Равномерное распределение.

2. Распределение Пуассона.

2 Теоретическая часть

2.1 Равномерное распределение

Случайная величина X имеет равномерное распределение на отрезке $[a; b]$, если ее функция плотности имеет вид

$$f(x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b \\ 0, & \text{иначе} \end{cases} \tag{2.1}$$

Обозначение: $X \sim R[a, b]$.

Функция равномерного распределения имеет следующий вид.

$$F(x) = \begin{cases} 0, & a < x \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & x > b \end{cases} \tag{2.2}$$

2.2 Распределение Пуассона

Случайная дискретная величина X распределена по закону Пуассона с параметром $\lambda > 0$, если она принимает значения 0, 1, 2, ... с вероятностями

$$P\left\{X = k\right\} = \frac{\lambda^k}{k!} * e^{-\lambda}, k \in 0, 1, 2, ..., \tag{2.3}$$

где

- k – количество событий,
- λ – математическое ожидание случайной величины.

Обозначение: $X \sim \Pi(\lambda)$.

Функция плотности распределения имеет вид:

$$P\left\{x = k\right\} = \frac{\lambda^k}{k!} * e^{-\lambda}, k \in 0, 1, 2, ... \tag{2.4}$$

Тогда соответствующая функция распределения имеет следующий вид.

$$F(x) = P\left\{X < x\right\}, X \sim \Pi(\lambda) \tag{2.5}$$

3 Результат

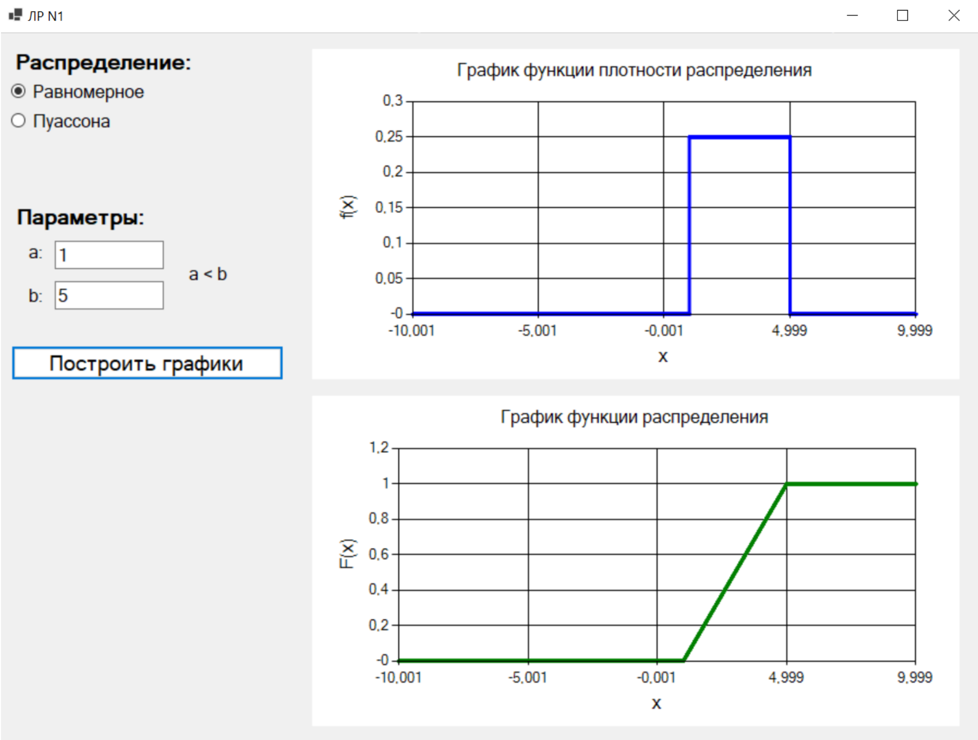

Рисунок 3.1 – Равномерное распределение случайной величины при $a = 1$, $b = 5$

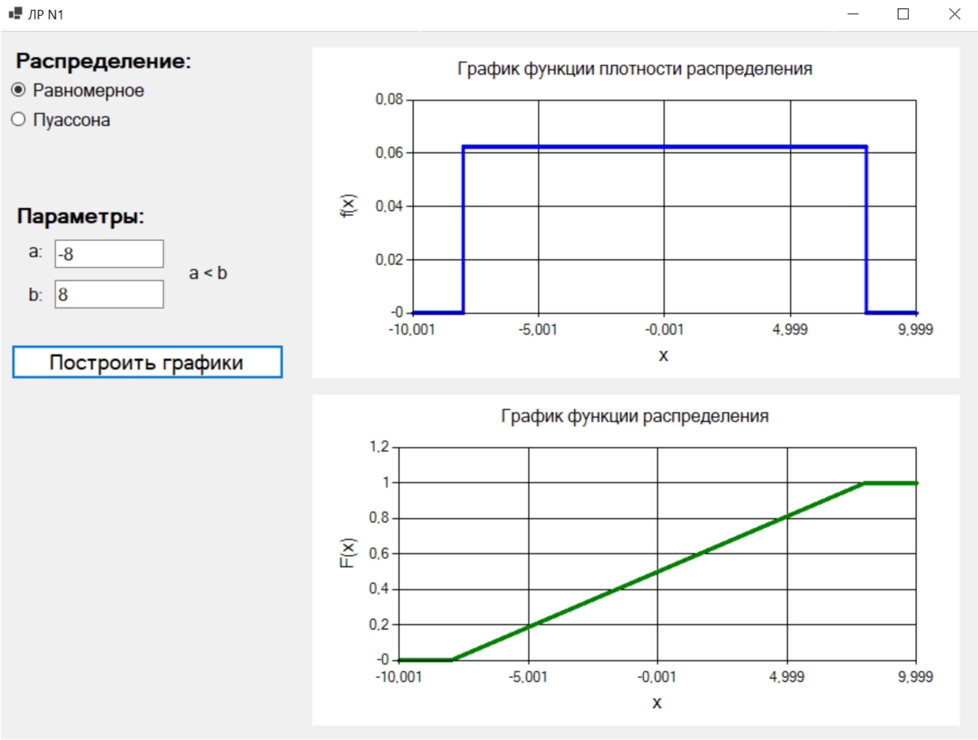

Рисунок 3.2 – Равномерное распределение случайной величины при $a = -8$, $b = 8$

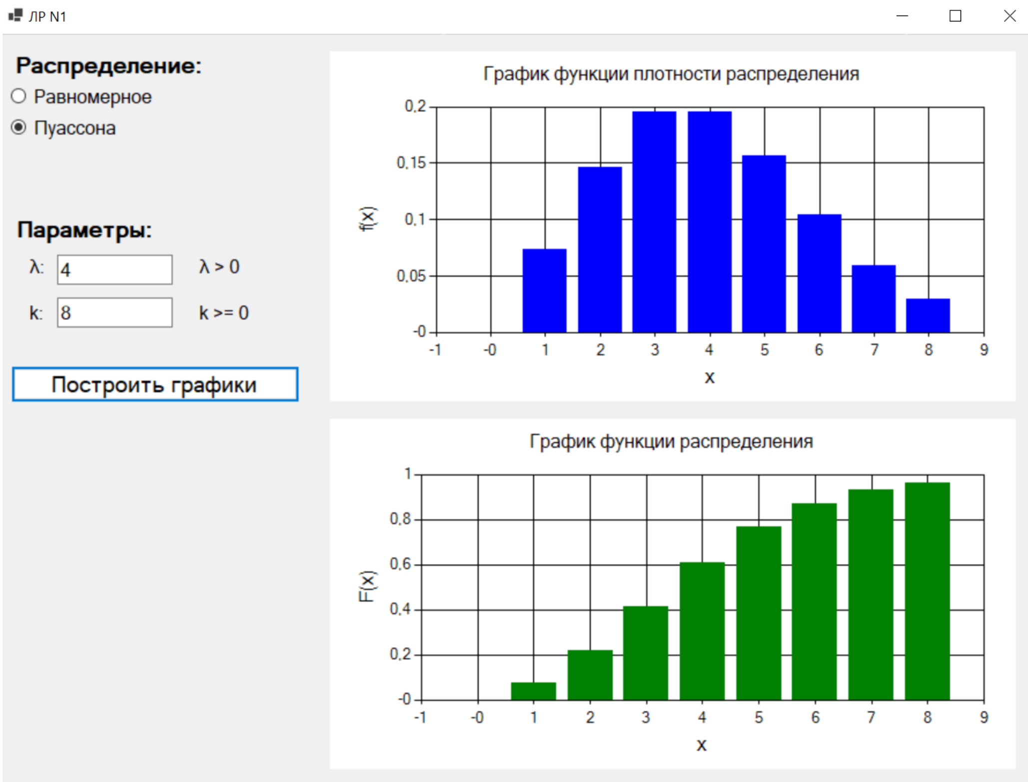

Рисунок 3.3 – Распределение случайной величины Пуассона при $\lambda = 4$, $k = 8$

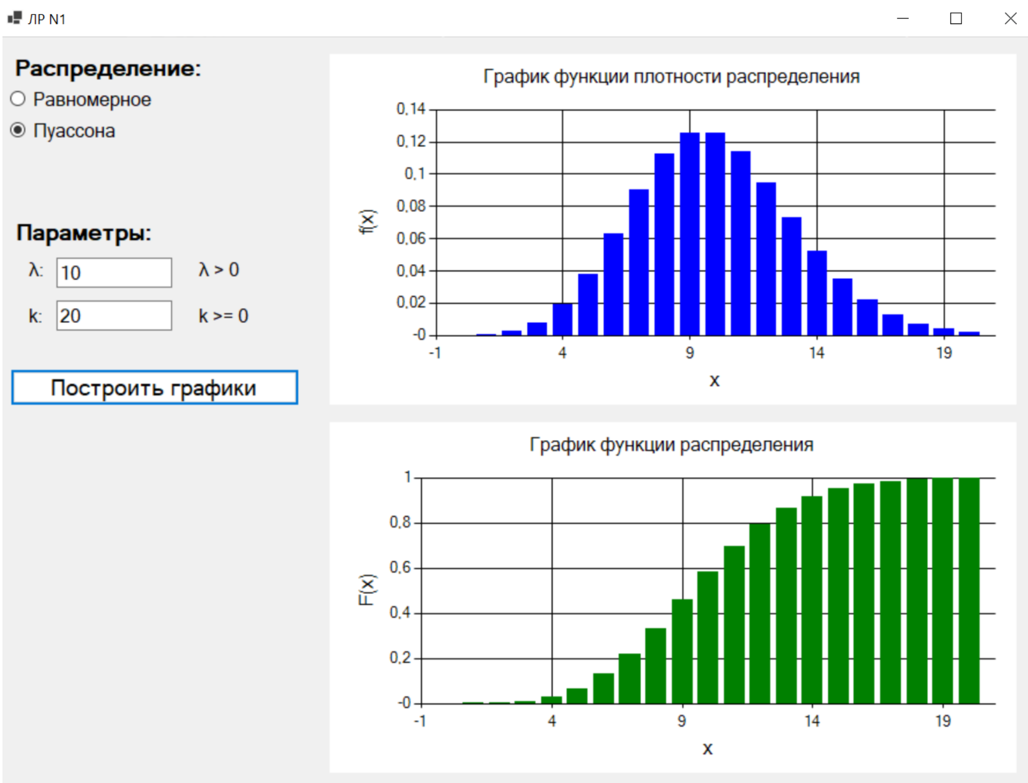

Рисунок 3.4 – Распределение случайной величины Пуассона при $\lambda = 10$, $k = 20$

4 Код программы

В листингах 4.1–4.2 представлен основной код программы.

Листинг 4.1 – Класс Distribution (часть 1)

```
public class Distribution
{
    public static void Uniform(double a, double b, double[] arrX,
        out double[] arrf, out double[] arrF)
    {
        int n = arrX.Length;
        arrf = new double[n];
        arrF = new double[n];

        if (a >= b)
            throw new Exception();

        for (int i = 0; i < n; i++)
        {
            arrf[i] = _Uniformf(a, b, arrX[i]);
            arrF[i] = _UniformF(a, b, arrX[i]);
        }
    }

    private static double _Uniformf(double a, double b, double x)
    {
        return (a <= x && x <= b) ? 1 / (b - a) : 0;
    }

    private static double _UniformF(double a, double b, double x)
    {
        if (x < a)
            return 0;

        if (x > b)
            return 1;

        return (x - a) / (b - a);
    }
```

Листинг 4.2 – Класс Distribution (часть 2)

```
    public static void Poisson(double lambda, int k, double[] arrX, out double[] arrf, out double[] arrF)
    {
        ulong n = (ulong)arrX.Length;
        arrf = new double[n];
        arrF = new double[n];

        if (k < 0 || lambda < 1e-4)
            throw new Exception();

        ulong max_k = (ulong)arrX[n - 1];
        // arrFactorial
        ulong [] factorialK = new ulong[max_k + 1];
        factorialK[0] = 1;

        for (ulong i = 1; i < max_k + 1; i++)
        {
            factorialK[i] = i * factorialK[i - 1];
        }

        ulong factk_tmp = 1;
        arrF[0] = 0;
        arrf[0] = _Poissonf(lambda, arrX[0], factk_tmp);

        for (ulong i = 1; i < n; i++)
        {
            factk_tmp =  (arrX[i] < 1e-4) ? 1: factorialK[(ulong)arrX[i]];
            arrf[i] = _Poissonf(lambda, arrX[i], factk_tmp);
            arrF[i] = arrF[i - 1] + arrf[i - 1];
        }
    }

    private static double _Poissonf(double lambda, double k, ulong factorial_k)
    {
        return k < 1e-4 ? 0 : (Math.Pow(lambda, k) / factorial_k) * Math.Exp(-lambda);
    }
}
```