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

КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

ГЕНЕРАЦИЯ ПРОИЗВОЛЬНЫХ РАСПРЕДЕЛЕНИЙ

отчет о
лабораторной работе №6

по дисциплине
ТЕХНОЛОГИИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ

ВАРИАНТ 2

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

Тула, 2022 г.

ЦЕЛЬ И ЗАДАЧА РАБОТЫ

Цель: изучение генерации случайных последовательностей по произвольно заданному закону распределения.

Задача: в данной работе требуется написать программу, демонстрирующую использование изученных принципов.

ЗАДАНИЕ НА РАБОТУ

Сгенерировать распределение, соответствующее рисунку 1:

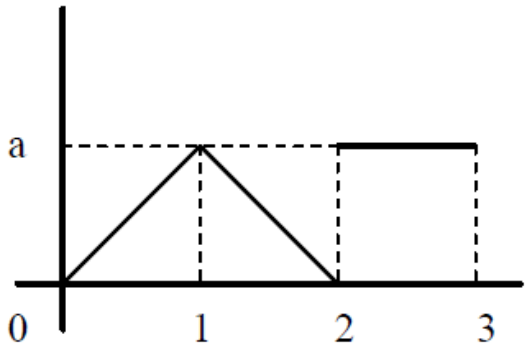

Рисунок 1 – График функции плотности вероятности

СХЕМА ПРОГРАММЫ

Схема алгоритма генерации произвольного распределения представлена на рисунке 2.

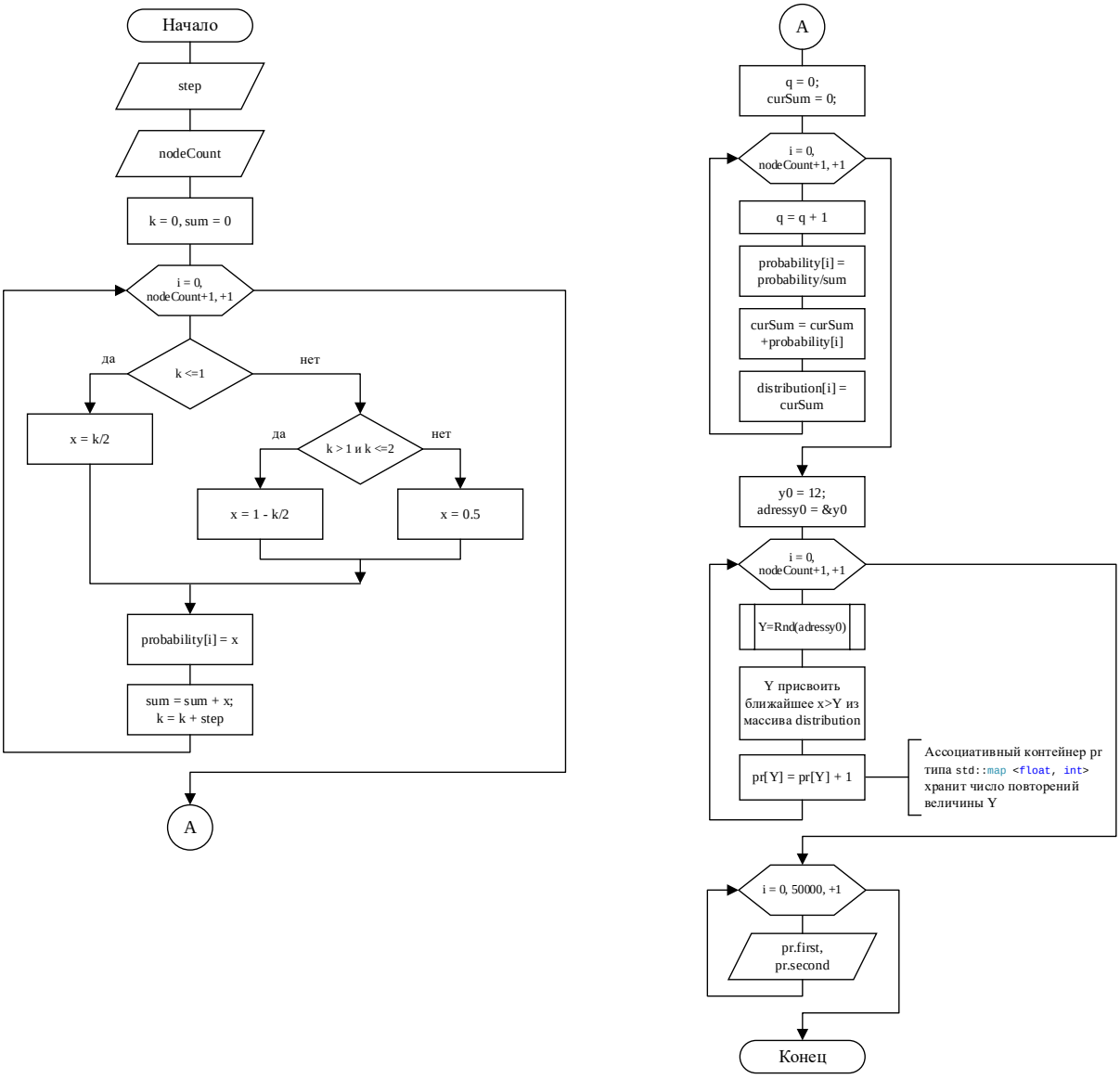

Рисунок 2 – Схема алгоритма генерации произвольного распределения

ТЕКСТ ПРОГРАММЫ

Текст программы на языке программирования C++ для генерации произвольного распределения представлен в листинге 1.

Листинг 1. Текст программы

```
#include <iostream>
#include <fstream>
#include <vector>
#include <map>
float 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");
 std::ofstream outX("outputX.txt");
 std::ofstream outY("outputY.txt");
 std::vector <float> probability;            //массив значений f(x)
                                             //в узлах сетки
 std::vector <float> distribution;           //массив значений F(x)
                                             //в узлах сетки
 float x;
 float step = 0.1;
 float nodeCount = 30;
 std::cout << "\n\t\t\t\tВведите длину шага: "; std::cin >> step;
 std::cout << "\n\t\t\t\tВведите количество узлов сетки: ";
std::cin >> nodeCount;

 float k = 0;
 float sum = 0;
 for (int i = 0; i < nodeCount + 1; i++)     //расчёт значений f(x)
                                             //в узлах сетки
 {
       if (k <= 1) x = k / 2;
       else {
             if (k > 1 && k <= 2) x = 1 - k / 2;
             else x = 0.5;
       }
       probability.push_back(x);
       sum += x;
       k += step;
 }

 int q = 0;
 float curSum = 0;
 for (auto to:probability)            //заполнение массива значениями F(x)
 {
       q++;
       to = to / sum;
       curSum += to;
       distribution.push_back(curSum);
 }

 float Y;
 int y0 = 12;
```

Листинг 1. Текст программы (продолжение)

```
 int* adressy0 = &y0;
 std::map <float, int> pr;              //ассоциативный контейнер для
                                        //подсчёта повторений числа
 std::vector<float>::iterator it;
 it = distribution.begin();             //итератор для работы с массивом


 for (int i = 0; i < 50000; i++)        //цикл генерации случайных величин

 {
       Y = Rnd(adressy0);
       it = upper_bound(distribution.begin(), distribution.end(), Y);
       Y = *it;
       pr[*it]++;
 }

 for (auto to : pr)                 //цикл для вывода значений в файл
                                    //для построения гистограммы
 {
       outX << to.first << "\n";              //вывод величины
       outY << (float)to.second / 50000       //вывод вероятности
                                              //соответствующей величины
              << "\n";
 }
 outX.close();
 outY.close();
 return 0;
}
```

ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ

Данная программа предназначена для генерации произвольного распределения. Пользователю предлагается ввести длину шага и количество узлов. После программа выполняет необходимые вычисления и выводит данные в файл.

ИНСТРУКЦИЯ ПРОГРАММИСТА

Данная программа предназначена для генерации произвольного распределения. Структуры данных, используемые в программе, приведены в таблице 1.

Таблица 1 – Структуры данных в программе

Имя	Тип (класс)	Предназначение
outX, outY	file	Имена файлов вывода
step	float	Шаг сетки
nodeCount	int	Число узлов в сетке
probability	std::vector <float>	Массив значений функции сти
distribution	std::vector <float>	Массив значений функции деления
x	float	Значение функции плотности
k	float	Счётчик пройденных шагов
sum	float	Сумма значений функции плотности
q	int	Счётчик
curSum	float	Локальная сумма значений функции деления
Y	float	Случайная величина
y0	int	Начальное значение для генератора
adressy0	*int	Адрес ячейки памяти y0
pr	std::map <float, int>	Ассоциативный контейнер для та повторений величины Y
it	std::vector<float>::iterator	Итератор для работы с контейнером

В программе имеются следующие функции:

1) `float Rnd(int* x0)` – линейный конгруэнтный генератор.

ДЕМОНСТРАЦИОННЫЙ ПРИМЕР

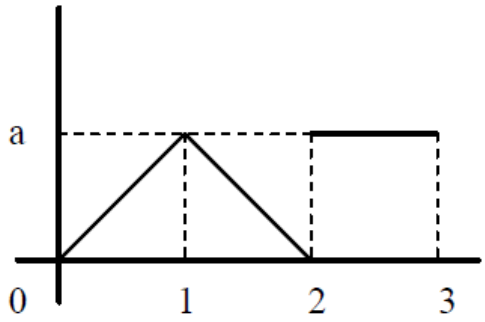

Рисунок 3 – Заданный график плотности вероятности

По данному графику функции плотности вероятности, определим постоянную a, используя условие нормировки $\int_{-\infty}^{\infty} f(x)dx = 1$. Поскольку интеграл является площадью под графиком, нетрудно посчитать, что a = 0.5. Теперь запишем саму функцию плотности вероятности, учтя константу a:

$$f(x) = \begin{cases} \frac{x}{2}, x \in [0,1] \\ 1 - \frac{x}{2}, x \in (1,2] \\ \frac{1}{2}, x \in (2,3] \end{cases}$$

Теперь можно задать массив, содержащий значения этой функции с приемлемым шагом дискретизации. Пусть шаг будет равен 0.1, а число узлов 30, тогда имеем следующий массив (таблица 1):

Таблица 1 – Значения $f(x)$ в узлах сетки с шагом 0.1

0	**0.05**	**0.1**	**0.15**	**0.2**	**0.25**	**0.3**	**0.35**	**0.4**	**0.45**
0.5	**0.45**	**0.35**	**0.25**	**0.2**	**0.15**	**0.1**	**0.05**	**0.5**	**0.5**
0.5	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**

Проведём нормировку этого массива, разделив каждое значение на сумму всех значений, равную 10.5. Теперь массив имеет вид (таблица 2):

Таблица 2 – Значения $f(x)$ в узлах сетки с шагом 0.1 после нормировки

0	**0.004**	**0.009**	**0.014**	**0.019**	**0.023**	**0.028**	**0.033**	**0.038**	**0.042**
0.047	**0.042**	**0.038**	**0.033**	**0.028**	**0.023**	**0.019**	**0.014**	**0.009**	**0.004**
0.047	**0.047**	**0.047**	**0.047**	**0.047**	**0.047**	**0.047**	**0.047**	**0.047**	**0.047**

Теперь, зная значения плотности вероятности $f(x)$ в каждом узле, можно таблично задать функцию распределения $F(x)$. Для этого используем смысл функции плотности распределения, которая показывает приращение функции распределения на каждом следующем ее шаге. По сути, таблица 2 задаёт значения производной $dF(x)$ в каждом узле. Таким образом, значения $F(x)$ на каждом следующем шаге мы получаем путем суммирования предыдущего значения со значением функции распределения в соответствующем узле. Сохраним полученные значения в виде массива (таблица 3):

Таблица 3 – Значения $F(x)$ в узлах сетки с шагом 0.1

0	**0.004**	**0.014**	**0.028**	**0.047**	**0.071**	**0.1**	**0.133**	**0.171**	**0.214**
0.261	**0.304**	**0.342**	**0.376**	**0.404**	**0.428**	**0.447**	**0.461**	**0.471**	**0.476**
0.523	**0.571**	**0.619**	**0.666**	**0.714**	**0.761**	**0.809**	**0.857**	**0.904**	**0.952**

Далее предположим, что случайная величина между узлами распределена равномерно и таким образом получим функцию распределения в виде непрерывной кусочной функции. Тогда для получения случайной величины с заданным законом распределения необходимо получить значение от генератора равномерного распределения (от 0 до 1) и, используя график функции распределения, получить случайную величину. График функции распределения представлен на рисунке 4.

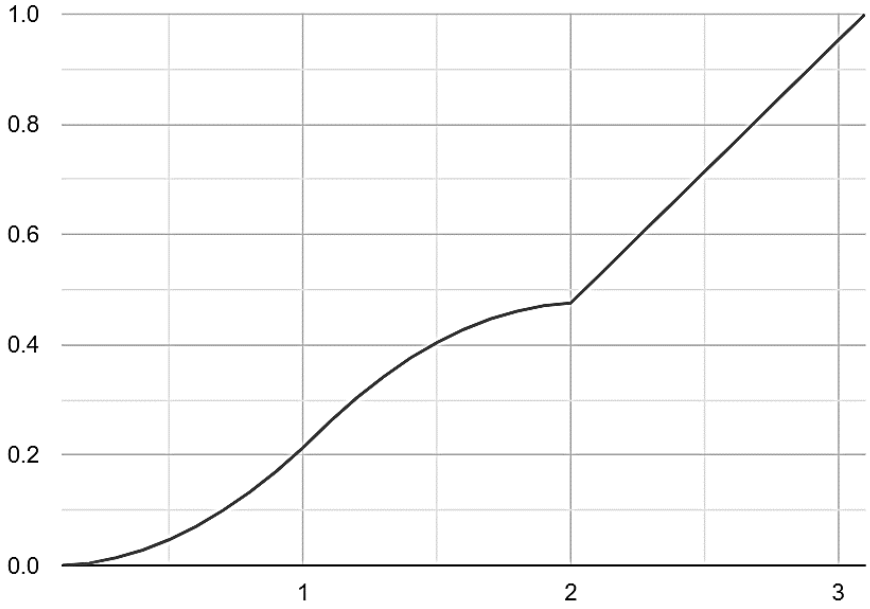

Рисунок 4 – График полученной кусочно-заданной функции распределения

С помощью полученной функции распределения необходимо найти последовательность из 50000 случайных величин и отобразить полученный результат в виде гистограммы. В программе возьмём шаг равный 0.05 и разобьём сетку на 200 узлов (рисунок 5):

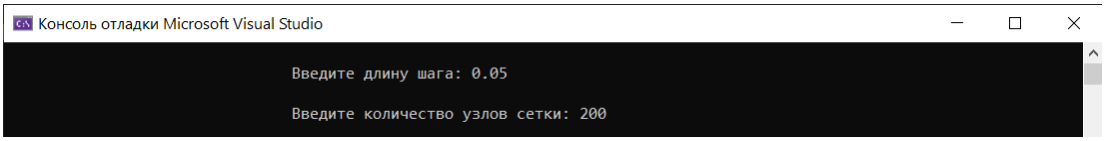

Рисунок 5 – Данные, вводимые в программу для генерации случайных величин

Содержимое файлов вывода (полученные случайные величины и их вероятности) представлены на рисунке 6.

outputY.txt – Блокнот

Файл Правка Формат Вид Справка

```
0.00026 0.00048 0.00088 0.00106 0.00136 0.00158 0.0019 0.0021 0.0025
0.00256 0.00286 0.00338 0.0035 0.0038 0.00412 0.0047 0.0046 0.00498
0.00528 0.00536 0.00528 0.00502 0.00458 0.00446 0.00428 0.0038 0.00356
0.00322 0.00318 0.00288 0.0024 0.0022 0.00184 0.00168 0.00156 0.00114
0.0009 0.00054 0.00022 0.00548 0.0058 0.0056 0.00546 0.00558 0.00556
0.00552 0.00548 0.00542 0.0057 0.00552 0.00558 0.00538 0.0056 0.0056
0.0058 0.00574 0.0057 0.00554 0.00562 0.00562 0.0054 0.00524 0.00568
0.00554 0.00568 0.00588 0.00562 0.00572 0.00592 0.00554 0.00536
0.00536 0.00558 0.00538 0.00556 0.00562 0.00532 0.0053 0.00538 0.00564
0.0057 0.00502 0.00552 0.00556 0.0058 0.00568 0.00568 0.0055 0.00552
0.00578 0.00556 0.00564 0.0056 0.0058 0.00556 0.00554 0.0056 0.00566
0.00554 0.0056 0.00534 0.00568 0.00544 0.00532 0.00564 0.0059 0.00542
0.00548 0.00558 0.0056 0.0055 0.00562 0.00556 0.00548 0.00528 0.00552
0.00584 0.0055 0.00584 0.00554 0.0057 0.0055 0.00582 0.00574 0.00536
0.00568 0.00562 0.0057 0.0052 0.0053 0.00528 0.00574 0.00584 0.00576
0.00564 0.0055 0.00566 0.0057 0.0053 0.00566 0.00574 0.0055 0.00546
0.00554 0.00532 0.00544 0.0054 0.0057 0.00562 0.00562 0.00546 0.00566
0.0057 0.0054 0.00578 0.00562 0.00568 0.00554 0.00556 0.00578 0.00554
0.00556 0.0057 0.00574 0.00544 0.00534 0.00566 0.00572 0.00568 0.00548
0.00536 0.00592 0.00576 0.0056 0.0057 0.00542 0.0052 0.00548 0.0054
0.0052 0.00562 0.0054 0.0055 0.00554 0.00546 0.00534 0.00576 0.00552
0.00548 0.00536 0.0057 0.00528 0.0056 0.00562 0.00538 0.00558 0.00588
0.0053 0.0056
```

Стр 1, стлб 1 | 100% | Windows (CRLF) | UTF-8

outputX.txt – Блокнот

Файл Правка Формат Вид Справка

```
0.000277778 0.000833333 0.00166667 0.00277778 0.00416667 0.00583333
0.00777778 0.01 0.0125 0.0152778 0.0183333 0.0216667 0.0252778
0.0291667 0.0333333 0.0377778 0.0425 0.0475 0.0527778 0.0583333
0.0636111 0.0686111 0.0733333 0.0777778 0.0819445 0.0858333 0.0894445
0.0927778 0.0958334 0.0986111 0.101111 0.103333 0.105278 0.106944
0.108333 0.109444 0.110278 0.110833 0.111111 0.116667 0.122222 0.127778
0.133333 0.138889 0.144445 0.15 0.155556 0.161111 0.166667 0.172222
0.177778 0.183333 0.188889 0.194445 0.2 0.205556 0.211111 0.216667
0.222222 0.227778 0.233333 0.238889 0.244445 0.25 0.255556 0.261111
0.266667 0.272222 0.277778 0.283333 0.288889 0.294445 0.3 0.305556
0.311111 0.316667 0.322222 0.327778 0.333334 0.338889 0.344445 0.35
0.355556 0.361111 0.366667 0.372223 0.377778 0.383334 0.388889 0.394445
0.4 0.405556 0.411112 0.416667 0.422223 0.427778 0.433334 0.438889
0.444445 0.450001 0.455556 0.461112 0.466667 0.472223 0.477778 0.483334
0.48889 0.494445 0.500001 0.505556 0.511112 0.516667 0.522223 0.527779
0.533334 0.53889 0.544445 0.550001 0.555556 0.561112 0.566668 0.572223
0.577779 0.583334 0.58889 0.594445 0.600001 0.605557 0.611112 0.616668
0.622223 0.627779 0.633334 0.63889 0.644446 0.650001 0.655557 0.661112
0.666668 0.672223 0.677779 0.683335 0.68889 0.694446 0.700001 0.705557
0.711112 0.716668 0.722224 0.727779 0.733335 0.73889 0.744446 0.750001
0.755557 0.761113 0.766668 0.772224 0.777779 0.783335 0.78889 0.794446
0.800002 0.805557 0.811113 0.816668 0.822224 0.827779 0.833335 0.83889
0.844446 0.850002 0.855557 0.861113 0.866668 0.872224 0.877779 0.883335
0.888891 0.894446 0.900002 0.905557 0.911113 0.916668 0.922224 0.92778
0.933335 0.938891 0.944446 0.950002 0.955557 0.961113 0.966669 0.972224
0.97778 0.983335 0.988891 0.994446 1
```

Стр 1, стлб 1 | 100% | Windows (CRLF) | UTF-8

Рисунок 6 – Содержимое файлов вывода

Полученная гистограмма представлена на рисунке 7.

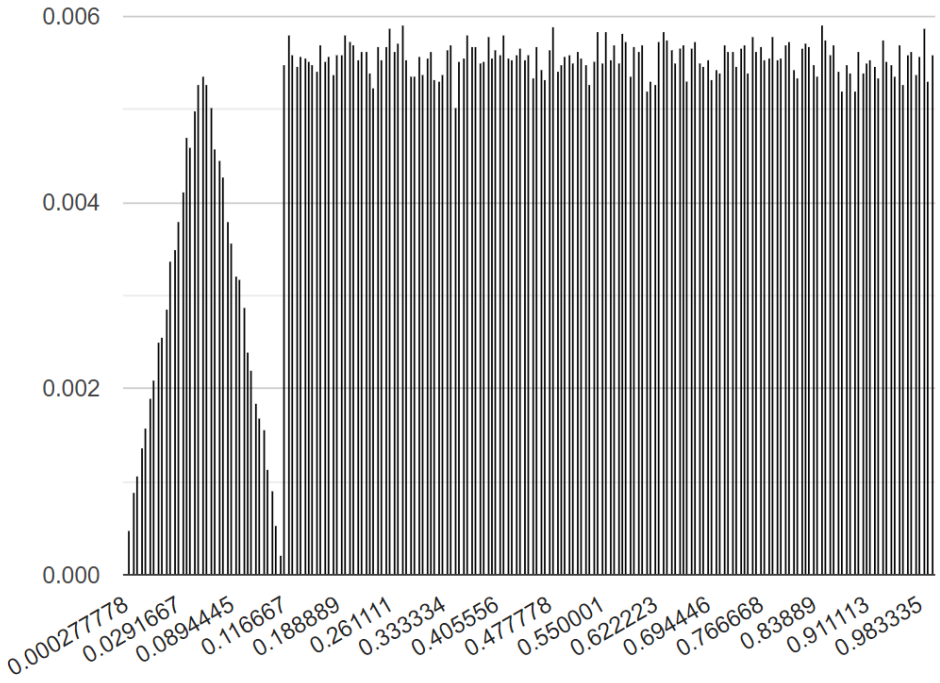

Рисунок 7 – График полученной гистограммы

Как видно по рисунку 7, гистограмма соответствует теоретической плотности распределения.

ВЫВОДЫ

В ходе данной лабораторной работы был изучен принцип работы сортировки слиянием по возрастанию двух файлов разного размера с использованием метода естественного двух путевого слияния. Такой подход обладает тем возможным преимуществом, что исходные файлы с преобладанием возрастающего или убывающего расположения элементов могут обрабатываться очень быстро, но при этом замедляется основной цикл вычислений.

Для демонстрации полученных знаний была написана программа для сортировки указанным методом, результат работы которой был проверен аналитически. По результатам проверки можно сделать вывод о том, что программа работает корректно.