**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**

**(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»**

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра 304 «Вычислительные машины, системы и сети»

Курсовой проект

По дисциплине «Программирование на языке высокого уровня»

По теме «Моделирование генератора случайных чисел с нормальным распределением»

Выполнил студент группы М30-225Бк-18

Мазаев А.Г.

Проверил доцент кафедры 304, к.т.н. Новиков П. В.

Москва 2019

Оглавление

[Теоретическая часть 3](#_Toc58527279)

[Блок схема программы 5](#_Toc58527280)

[Текст программы 14](#_Toc58527281)

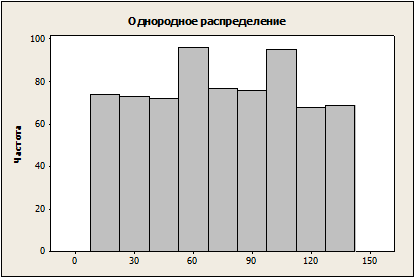
[Результаты работы программы 33](#_Toc58527282)

[Описание команд и идентификаторов библиотеки glut.h 38](#_Toc58527283)

[Список используемой литературы 39](#_Toc58527284)

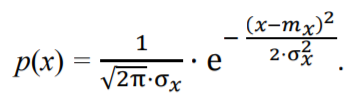
# Теоретическая часть

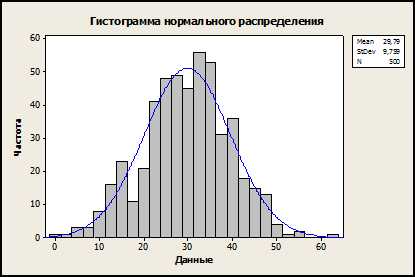
Гистограммы с большим количеством пиков (многомодальные) встречаются крайне редко и, зачастую свидетельствуют о присутствии специальных факторов, влияющих на исследуемую систему или процесс. Если каждый интервал гистограммы содержит примерно равное количество значений, то такая гистограмма называется однородной или гистограммой равномерного распределения:



Нормальному закону распределения может подчиняться любая величина, на которую не влияют специальные факторы (например, связывающие или ограничивающие): когда она подвержена влиянию большого числа случайных помех. Считается, что из всех распределений чаще всего встречается именно нормальное.

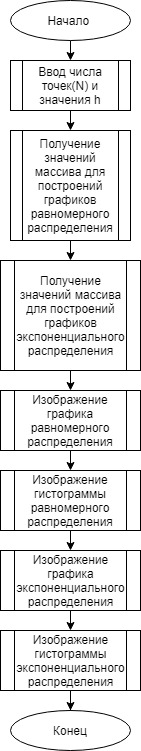
Нормальное (гауссово) распределение задаётся плотностью распределения:

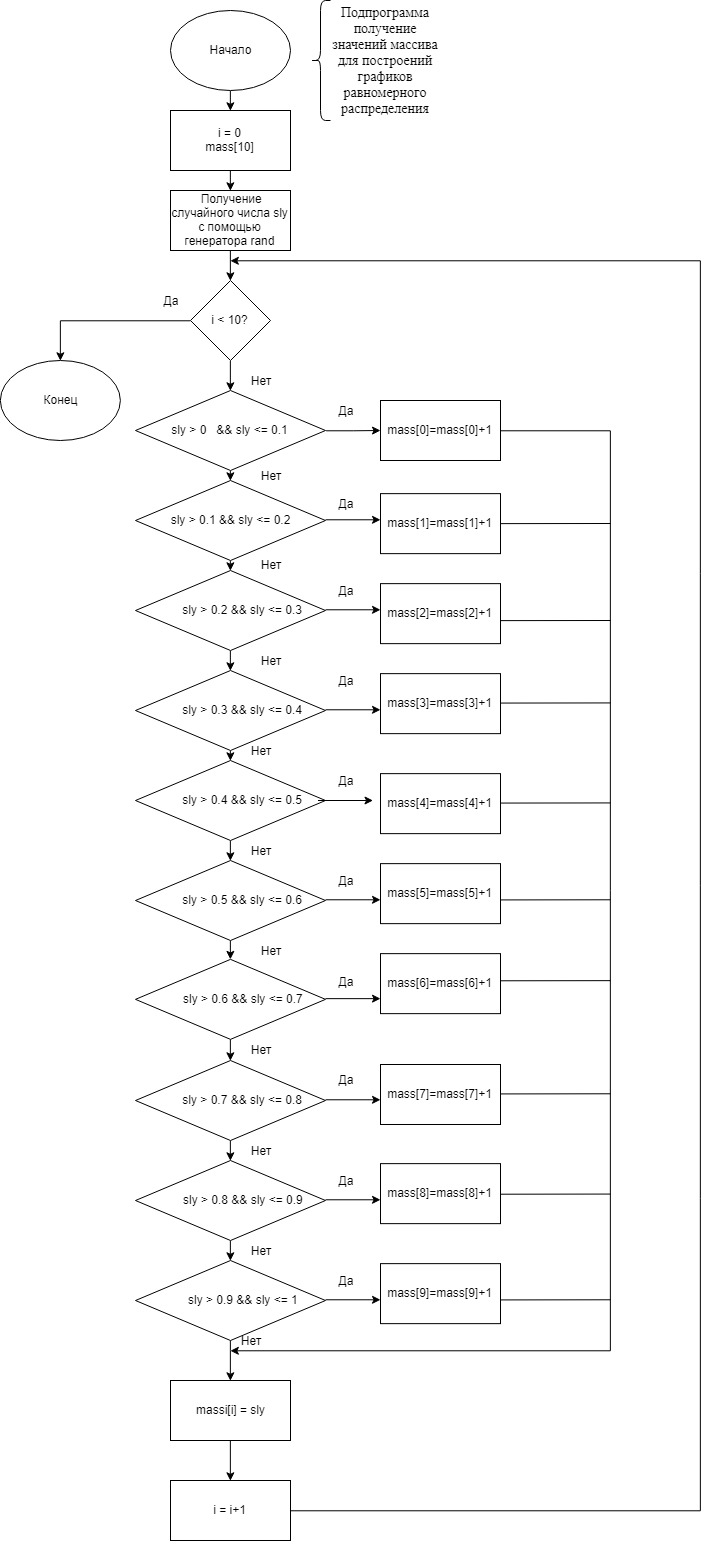


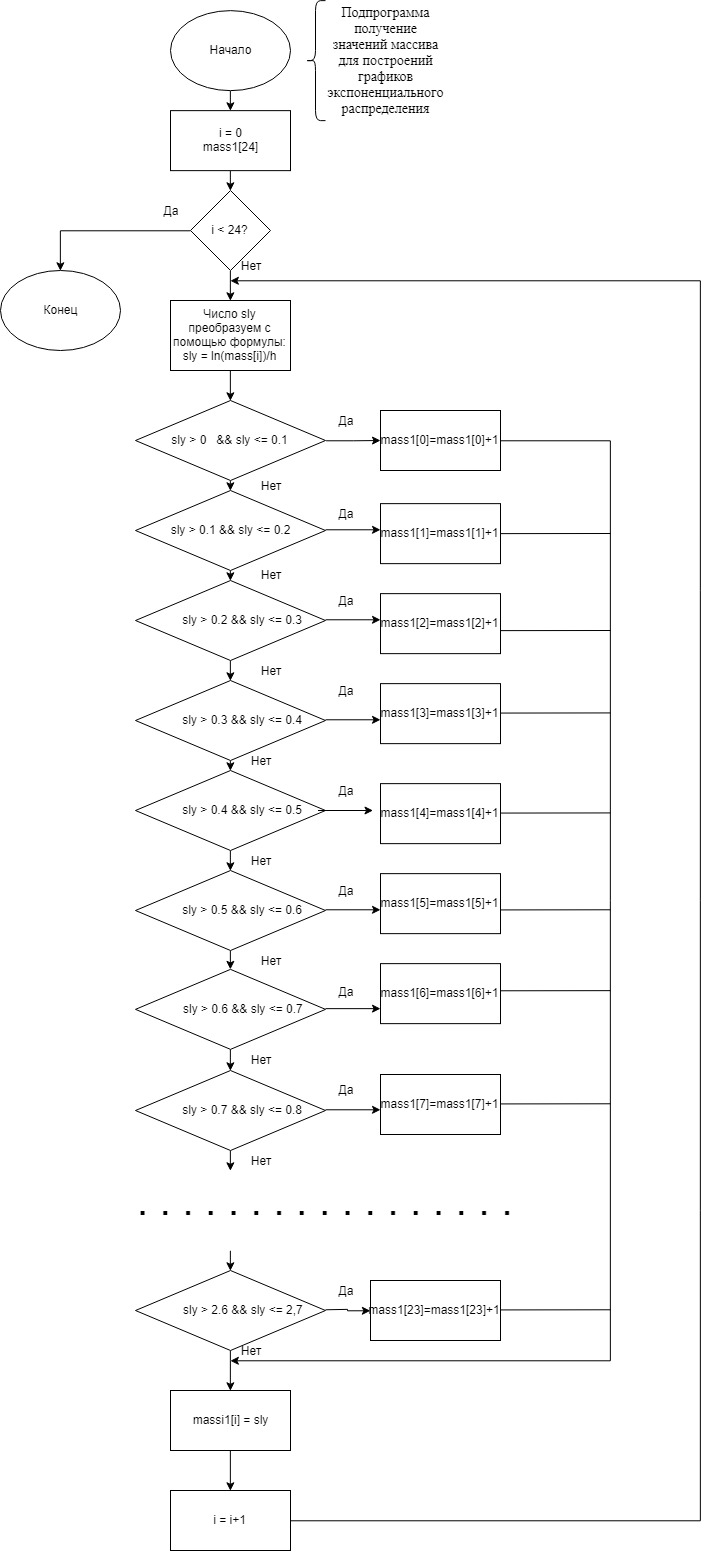


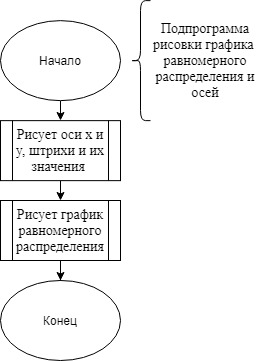
В этой работе было создается некоторое количество случайных значений(пользователь задает количество) от 0 до 1 с помощью функции random. При равномерном распределении считается кол-во значений попавших в определенный промежуток (0-1,1-2, …). При нормальном распределении случайное проходит через формулу плотности нормального распределения и потом так же считается количество попавших точке в определенный промежуток.

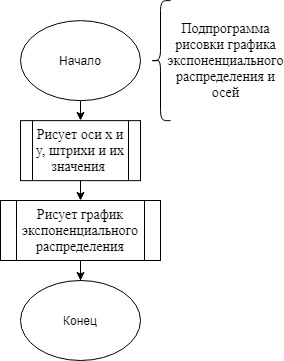
# Блок схема программы

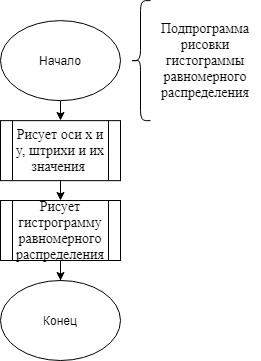






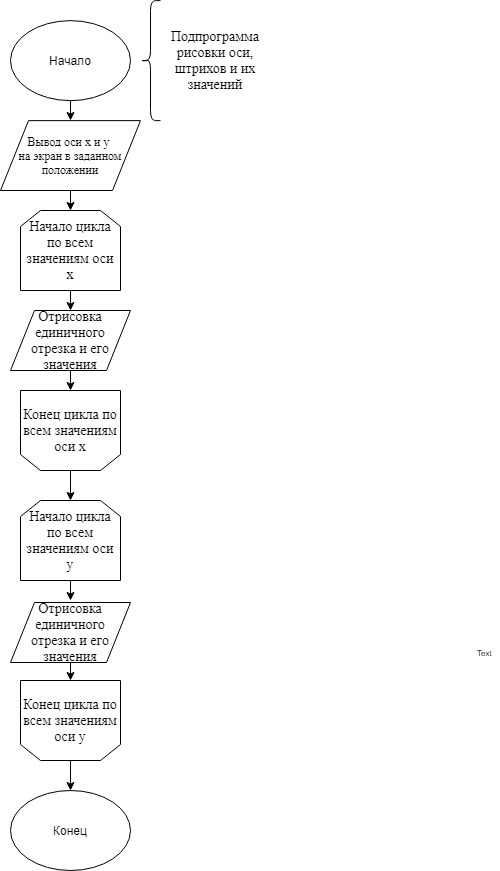




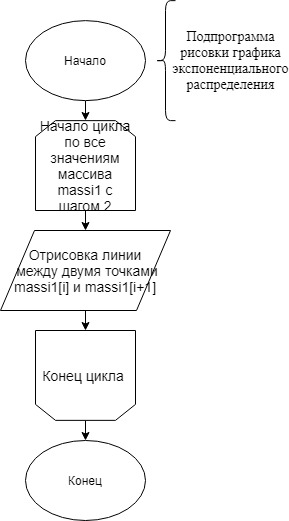












# Текст программы

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stdio.h"

#include "iostream"

#include "stdlib.h"

#include "math.h"

#include <algorithm>

#include <time.h>

#include <glut.h>

#include <string>

#include <fstream>

#include <windows.h>

using namespace std;

const double PI = 3.14159265;

int width = 1100;

int height = 600;

void DrawAxis();

void DrawAxis1();

void DrawStrip();

void DrawStrip1();

void Draw();

void Draw1();

void Enable2D(int width, int height);

void PrintText(float x, float y, string string);

void points();

void points1();

double\* massi;

double\* massi1;

int mass[10];

int mass1[30];

float N;

int Random(int min, int max) {// Возвращает псевдослучайное число из диапазона [min, max)

return min + rand() % (max - min);

}

void Init() // Инициализирует пространство

{

Enable2D(width, height);

glClearColor(1, 1, 1, 1);

}

void enter(float& N)

{

setlocale(LC\_ALL, "RUS");

cout << "Введите N: ";

cin >> N;

N = N \* 2;

}

void Draw()

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

DrawAxis(); //Рисует оси, значения и штрихи

DrawStrip(); //Рисует график

glutSwapBuffers();// Swap buffers (has to be done at the end)

}

void Draw1()

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

DrawAxis1();

DrawStrip1();

// Swap buffers (has to be done at the end)

glutSwapBuffers();

}

void DrawStrip()

{

glPointSize(2);

glBegin(GL\_LINE\_STRIP);

glColor3f(0.0f, 0.0f, 0.0f);

for (int i = 0; i + 3 < N; i += 2)

{

glVertex2f(2 \* massi[i] - width / 2 + 100, 400 \* massi[i + 1] - 200);

glVertex2f(2 \* massi[i + 2] - width / 2 + 100, 400 \* massi[i + 3] - 200);

}

glEnd();

}

void DrawStrip1() {

glPointSize(2);

glBegin(GL\_LINE\_STRIP);

glColor3f(0.0f, 0.0f, 0.0f);

for (int i = 0; i + 3 < N; i += 2)

{

glVertex2f(2 \* massi1[i] - width / 2 + 100, 100 \* massi1[i + 1] );

glVertex2f(2 \* massi1[i + 2] - width / 2 + 100, 100 \* massi1[i + 3]);

}

glEnd();

}

void points()

{

double sly;

massi = new double[N];

for (int i = 0, k = 0; i + 1 < N; i += 2, k++) {

sly = Random(0, 100) / 100.0;;

if (sly > 0 && sly <= 0.1) mass[0]++;

if (sly > 0.1 && sly <= 0.2) mass[1]++;

if (sly > 0.2 && sly <= 0.3) mass[2]++;

if (sly > 0.3 && sly <= 0.4) mass[3]++;

if (sly > 0.4 && sly <= 0.5) mass[4]++;

if (sly > 0.5 && sly <= 0.6) mass[5]++;

if (sly > 0.6 && sly <= 0.7) mass[6]++;

if (sly > 0.7 && sly <= 0.8) mass[7]++;

if (sly > 0.8 && sly <= 0.9) mass[8]++;

if (sly > 0.9 && sly <= 1) mass[9]++;

massi[i] = k;

massi[i + 1] = sly;

}

}

void points1()

{

float v\_k, x\_k;

massi1 = new double[N\*2];

float sly\_e, max = 0;

double r2;

srand(time(NULL));

int p =0 , m = 0;

for (int i = 1, k = 0, j = 0; i+1 < N\*2; i += 2, k += 2, j++) {

r2 = Random(0, 100) / 100.0;

//cout << r2 << " "<< massi[i] <<endl;

sly\_e = sin(2\*PI\*massi[i])\*pow(-2\*log(r2),0.5);

//cout << sly\_e << endl;

if (sly\_e < -0.9 && sly\_e >= -1) mass1[0]++;

if (sly\_e < -0.8 && sly\_e >= -0.9) mass1[1]++;

if (sly\_e < -0.7 && sly\_e >= -0.8) mass1[2]++;

if (sly\_e < -0.6 && sly\_e >= -0.7) mass1[3]++;

if (sly\_e < -0.5 && sly\_e >= -0.6) mass1[4]++;

if (sly\_e < -0.4 && sly\_e >= -0.5) mass1[5]++;

if (sly\_e < -0.3 && sly\_e >= -0.4) mass1[6]++;

if (sly\_e < -0.2 && sly\_e >= -0.3) mass1[7]++;

if (sly\_e < -0.1 && sly\_e >= -0.2) mass1[8]++;

if (sly\_e < 0 && sly\_e >= -0.1) mass1[9]++;

if (sly\_e > 0 && sly\_e <= 0.1) mass1[10]++;

if (sly\_e > 0.1 && sly\_e <= 0.2) mass1[11]++;

if (sly\_e > 0.2 && sly\_e <= 0.3) mass1[12]++;

if (sly\_e > 0.3 && sly\_e <= 0.4) mass1[13]++;

if (sly\_e > 0.4 && sly\_e <= 0.5) mass1[14]++;

if (sly\_e > 0.5 && sly\_e <= 0.6) mass1[15]++;

if (sly\_e > 0.6 && sly\_e <= 0.7) mass1[16]++;

if (sly\_e > 0.7 && sly\_e <= 0.8) mass1[17]++;

if (sly\_e > 0.8 && sly\_e <= 0.9) mass1[18]++;

if (sly\_e > 0.9 && sly\_e <= 1) mass1[19]++;

massi1[k] = j;

massi1[k + 1] = sly\_e;

}

}

void DrawAxis()// Оси на первом графике

{

// Draw X axis

glBegin(GL\_LINES);

glColor3f(0.0f, 0.0f, 0.0f);

glVertex2f(-width / 2, -200);

glVertex2f(width / 2, -200);

glEnd();

// Draw Y axis

glBegin(GL\_LINES);

glColor3f(0.0f, 0.0f, 0.0f);

glVertex2f(-width / 2 + 100, -height / 2);

glVertex2f(-width / 2 + 100, height / 2);

glEnd();

// стрелка Y

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);

glBegin(GL\_TRIANGLES);

glColor3f(0, 0, 0);

glVertex2f(-width / 2 + 100, height / 2);

glVertex2f(-width / 2 + 100 - 5, height / 2 - 5);

glVertex2f(-width / 2 + 100 + 5, height / 2 - 5);

// стрелка X

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);

glBegin(GL\_TRIANGLES);

glColor3f(0, 0, 0); // рисуем треугольник

glVertex2f(width / 2, 0);

glVertex2f(width / 2 - 5, -205);

glVertex2f(width / 2 - 5, -195);

glEnd();

PrintText(-25 - width / 2 + 100, height / 2 - 15, "v(x)");

PrintText(width / 2 - 18, -190, "x");

// штрихи на х

glLineWidth(1); // ширина линии 1

glBegin(GL\_LINES);

glColor3d(0, 0, 0); // цвет

for (int i = -width / 2; i < width / 2; i += 20) {

glVertex2f(i, -202);

glVertex2f(i, -198);

}

glEnd();

//значение на х

glColor3f(0, 0, 0);

char values\_1[5] = "";

for (int i = -width / 2 + 100, k = 0; i < height; i += 100, k += 50) {

if (k != 0)

{

glRasterPos2f(i - 5, -215);

sprintf\_s(values\_1, "%d", k);

int j = strlen(values\_1);

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

{

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, values\_1[i]);

}

}

}

int step = 20;

int stepy = 40;

// штрихи на y

glLineWidth(1); // ширина линии 1

glBegin(GL\_LINES);

glColor3d(0, 0, 0); // цвет

for (int i = -height / 2 + 20; i < height / 2; i += stepy) {

glVertex2f(-width / 2 + 94, i);

glVertex2f(-width / 2 + 106, i);

}

glEnd();

//значения на y

glColor3f(0, 0, 0);

char values[5] = "";

for (int i = 0, k = 0; k <= 10; i += stepy, k++) {

sprintf\_s(values, "%d", k);

if (k == 10)PrintText(-width / 2 + 70, i - 205, "1");

else {

PrintText(-width / 2 + 70, i - 205, "0,");

int j = strlen(values);

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

{

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, values[i]);

}

}

}

}

void DrawAxis1()//оси на втором графике

{

// Draw X axis

glBegin(GL\_LINES);

glColor3f(0.0f, 0.0f, 0.0f);

glVertex2f(-width / 2, 0);

glVertex2f(width / 2, 0);

glEnd();

// Draw Y axis

glBegin(GL\_LINES);

glColor3f(0.0f, 0.0f, 0.0f);

glVertex2f(-width / 2 + 100, -height / 2);

glVertex2f(-width / 2 + 100, height / 2);

glEnd();

// стрелка Y

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);

glBegin(GL\_TRIANGLES);

glColor3f(0, 0, 0);

glVertex2f(-width / 2 + 100, height / 2);

glVertex2f(-width / 2 + 100 - 5, height / 2 - 5);

glVertex2f(-width / 2 + 100 + 5, height / 2 - 5);

// стрелка X

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);

glBegin(GL\_TRIANGLES);

glColor3f(0, 0, 0); // рисуем треугольник

glVertex2f(width / 2, 0);

glVertex2f(width / 2 - 5, -5);

glVertex2f(width / 2 - 5, 5);

glEnd();

PrintText(-25 - width / 2 + 100, height / 2 - 15, "v(x)");

PrintText(width / 2 - 18, 10, "x");

// штрихи на х

glLineWidth(1); // ширина линии 1

glBegin(GL\_LINES);

glColor3d(0, 0, 0); // цвет

for (int i = -width / 2; i < width / 2; i += 20) {

glVertex2f(i, -2);

glVertex2f(i, 2);

}

glEnd();

//значение на х

glColor3f(0, 0, 0);

char values\_1[5] = "";

for (int i = -width / 2 + 100, k = 0; i < height; i += 100, k += 50) {

if (k != 0)

{

glRasterPos2f(i - 5, -15);

sprintf\_s(values\_1, "%d", k);

int j = strlen(values\_1);

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

{

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, values\_1[i]);

}

}

}

int step = 20;

int stepy = 25;

glLineWidth(1); // Ширина линии 1

glBegin(GL\_LINES);

glColor3d(0, 0, 0);

for (int i = -height / 2; i < height / 2; i += stepy) // Штрихи на y

{

glVertex2f(-width / 2 + 94, i);

glVertex2f(-width / 2 + 106, i);

}

glEnd();

glColor3f(0, 0, 0); // Значения на y

PrintText(-width / 2 + 70, -25, "-0,1");

PrintText(-width / 2 + 70, -50, "-0,2");

PrintText(-width / 2 + 70, -75, "-0,3");

PrintText(-width / 2 + 70, -100, "-0,4");

PrintText(-width / 2 + 70, -125, "-0,5");

PrintText(-width / 2 + 70, -150, "-0,6");

PrintText(-width / 2 + 70, -175, "-0,7");

PrintText(-width / 2 + 70, -200, "-0,8");

PrintText(-width / 2 + 70, -225, "-0,9");

PrintText(-width / 2 + 70, -250, "-1,0");

PrintText(-width / 2 + 70, 0, "0,0");

PrintText(-width / 2 + 70, 25, "0,1");

PrintText(-width / 2 + 70, 50, "0,2");

PrintText(-width / 2 + 70, 75, "0,3");

PrintText(-width / 2 + 70, 100, "0,4");

PrintText(-width / 2 + 70, 125, "0,5");

PrintText(-width / 2 + 70, 150, "0,6");

PrintText(-width / 2 + 70, 175, "0,7");

PrintText(-width / 2 + 70, 200, "0,8");

PrintText(-width / 2 + 70, 225, "0,9");

PrintText(-width / 2 + 70, 250, "1,0");

}

void Enable2D(int width, int height)

{

glViewport(0, 0, width, height);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-width / 2.0f, width / 2.0f, -height / 2.0f, height / 2.0f, 0.0f, 1.0f);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void gistogramma(void) // Рисует 1 гистограмму

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

glBegin(GL\_LINES); //Х

glColor3f(0.0f, 0.0f, 0.0f);

glVertex2f(-width / 2, -200);

glVertex2f(width / 2, -200);

glEnd();

glBegin(GL\_LINES);//Y

glColor3f(0.0f, 0.0f, 0.0f);

glVertex2f(-width / 2 + 85, -height / 2);

glVertex2f(-width / 2 + 85, height / 2);

glEnd();

glBegin(GL\_LINES); // Штрихи

glLineWidth(1);

glColor3d(0, 0, 0);

for (int i = -height / 2 + 100; i < height / 2; i += 100) {

glVertex2f(-width / 2 + 78, i);

glVertex2f(-width / 2 + 90, i);

}

glEnd();

PrintText(-50 - width / 2 + 100, height / 2 - 15, "p(x)");

PrintText(width / 2 - 18, -190, "x");

char values[5] = "";

for (int i = -height / 2 + 100, k = 0; i < height / 2; i += 100, k += 500) //значения на у

{

glRasterPos2f(-width / 2 + 50, i);

sprintf\_s(values, "%d", k);

int j = strlen(values);

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

{

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, values[i]);

}

}

glEnd();

int step = 30;

char values1[5] = "";

//знаяения по х

glColor3f(0, 0, 0);

for (int i = (-width / 2) + 80, k = 0; k <= 10; i += step, k++) {

glRasterPos2f(i - 5, -215);

sprintf\_s(values1, "%d", k);

if (k == 10)PrintText(i - 5, -215, "1");

else {

PrintText(i - 5, -215, "0,");

int j = strlen(values1);

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

{

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, values1[i]);

}

}

}

glLineWidth(29);// Строим столбцы

glBegin(GL\_LINES);

glColor3d(0, 0, 0);

for (int i = (-width / 2) + 100, k = 0; k < N / 2 && k < 10; i += step, k++) {

glVertex2f(i, -200);

glVertex2f(i, -200 + mass[k]/5);

}

glEnd();

glutSwapBuffers();

}

void gistogramma1(void) // Рисует 1 гистограмму

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

glBegin(GL\_LINES); //Х

glColor3f(0.0f, 0.0f, 0.0f);

glVertex2f(-width / 2, -200);

glVertex2f(width / 2, -200);

glEnd();

glBegin(GL\_LINES);//Y

glColor3f(0.0f, 0.0f, 0.0f);

glVertex2f(-165, -height / 2);

glVertex2f(-165, height / 2);

glEnd();

glBegin(GL\_LINES); // Штрихи

glLineWidth(1);

glColor3d(0, 0, 0);

for (int i = -height / 2 + 100; i < height / 2; i += 100) {

glVertex2f(-170, i);

glVertex2f(-160, i);

}

glEnd();

PrintText(-160, height / 2 - 15, "p(x)");

PrintText(width / 2 - 18, -190, "x");

char values[5] = "";

for (int i = -height / 2 + 100, k = 0; i < height / 2; i += 100, k += 500) //значения на у

{

glRasterPos2f(-200, i);

sprintf\_s(values, "%d", k);

int j = strlen(values);

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

{

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, values[i]);

}

}

glEnd();

int step = 30;

glLineWidth(29);

glBegin(GL\_LINES);

glColor3d(0, 0, 0);

for (int i = (-width / 2) + 100, k = 0; k < 30; i += step, k++) {

glVertex2f(i, -200);

glVertex2f(i, -200 + mass1[k]/5);

}

glEnd();

glColor3f(0, 0, 0); //знаяения по х

PrintText(-470, -215, "-1,0");

PrintText(-440, -215, "-0,9");

PrintText(-410, -215, "-0,8");

PrintText(-380, -215, "-0,7");

PrintText(-350, -215, "-0,6");

PrintText(-320, -215, "-0,5");

PrintText(-290, -215, "-0,4");

PrintText(-260, -215, "-0,3");

PrintText(-230, -215, "-0,2");

PrintText(-200, -215, "-0,1");

PrintText(-160, -215, "0,1");

PrintText(-130, -215, "0,2");

PrintText(-100, -215, "0,3");

PrintText(-70, -215, "0,4");

PrintText(-40, -215, "0,5");

PrintText(-10, -215, "0,6");

PrintText(20, -215, "0,7");

PrintText(50, -215, "0,8");

PrintText(80, -215, "0,9");

PrintText(110, -215, "1,0");

glEnd();

glutSwapBuffers();

}

//Подпрограмма, показывающая текст на экране (для расположения координат и названий осей графиков)

void PrintText(float x, float y, string string)

{

char\* text = new char[string.length()];

glColor3f(0, 0, 0);

glRasterPos2f(x, y);

for (int i = 0; i < string.length(); i++)

text[i] = string[i];

for (int i = 0; i < string.length(); i++)

{

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, text[i]);

x += 2;

}

delete[] text;

}

int main(int argc, char\*\* argv)

{

clock\_t start, stop;

enter(N);

points();

points1();

// Initialize opengl (via glut)

glutInit(&argc, argv); // инициализация GLUT

glutInitWindowPosition(0, 300);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(width, height);// окно размера width х height

glutCreateWindow("Axes");//создание окна

glutDisplayFunc(Draw);

// регистрация обратных вызовов

Init();

glutInitWindowPosition(300, 300);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(width, height);// окно размера width х height

glutCreateWindow("Axes1");//создание окна

glutDisplayFunc(Draw1);

// регистрация обратных вызовов

Init();

glutInitWindowPosition(0, 0);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

/\* окно в цветовом пространстве RGB , с двойной буферизацией, с использованием буфера глубины\*/

glutInitWindowSize(width, height);// окно размера width х height

glutCreateWindow("gistogramma"); //

glutDisplayFunc(gistogramma);

Init();

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowPosition(400, 0);

glutInitWindowSize(width, height);// окно размера width х height

glutCreateWindow("gistogramma1"); //

glutDisplayFunc(gistogramma1);

Init();

// Основной цикл GLUT

cout << "time = " << clock() << endl;

glutMainLoop();

delete[] massi;

delete[] massi1;

return 0;

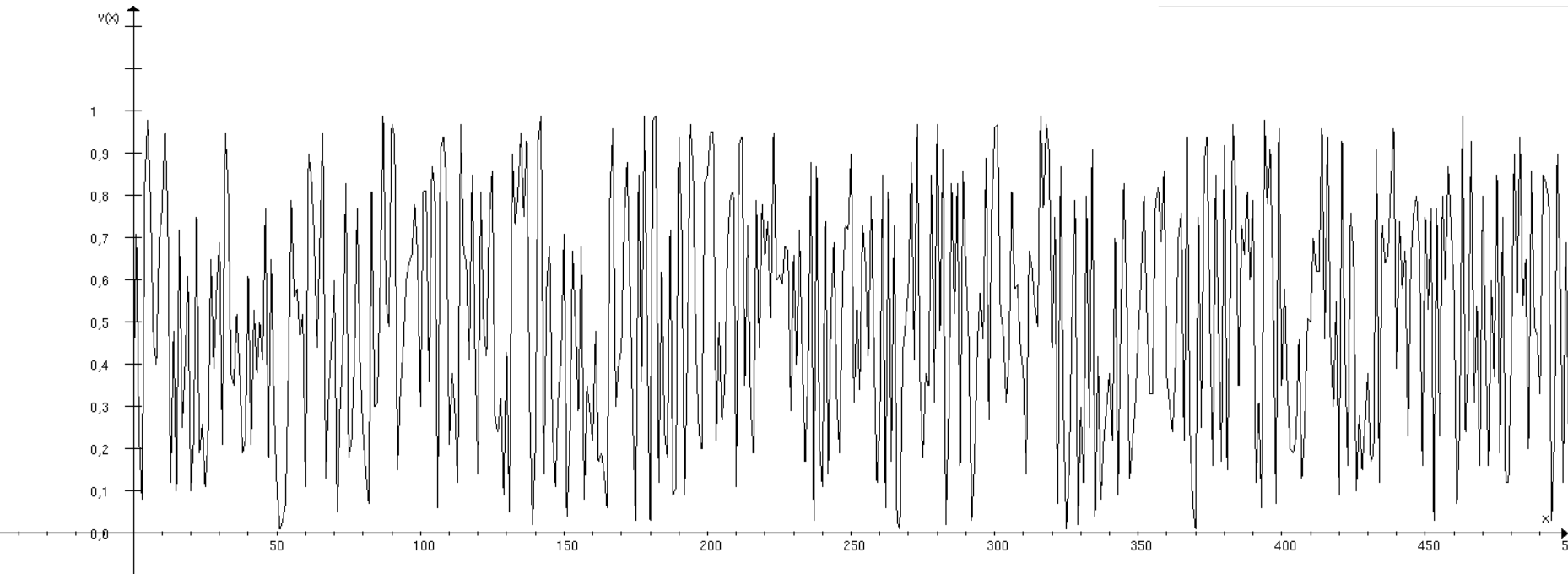
}

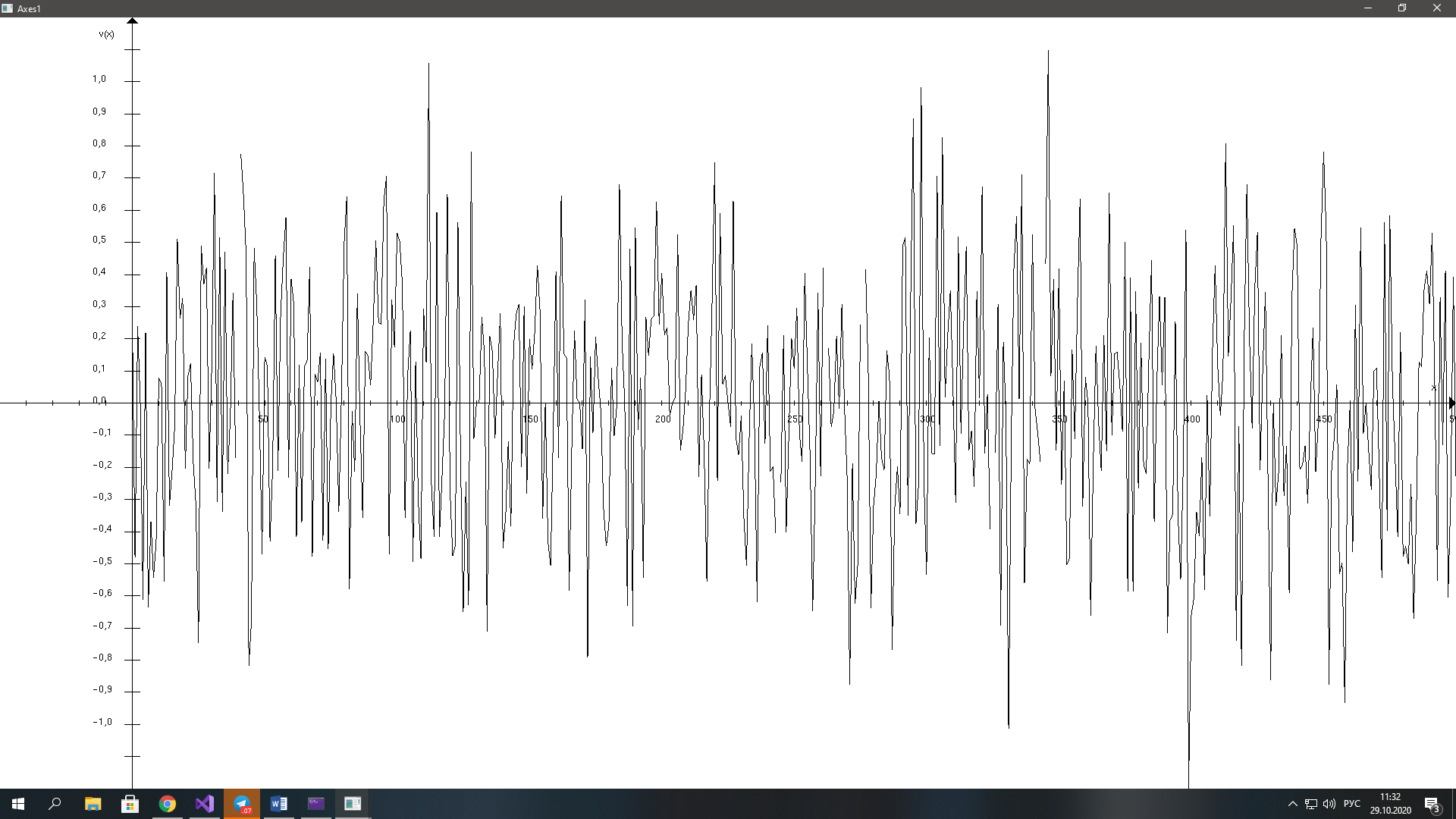
# Результаты работы программы

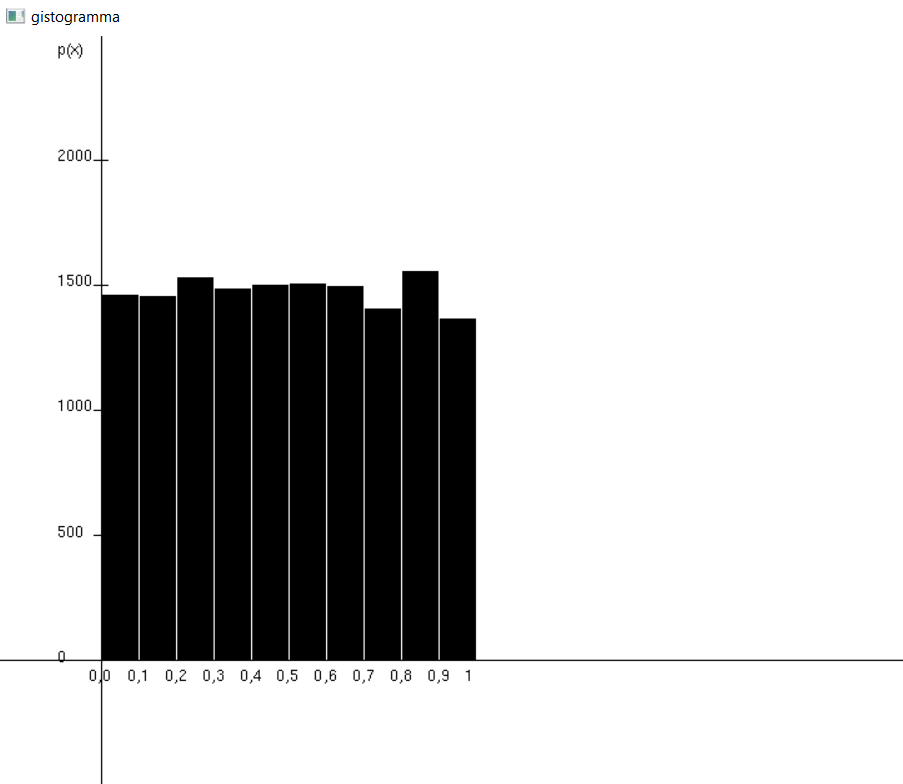
При входных данных

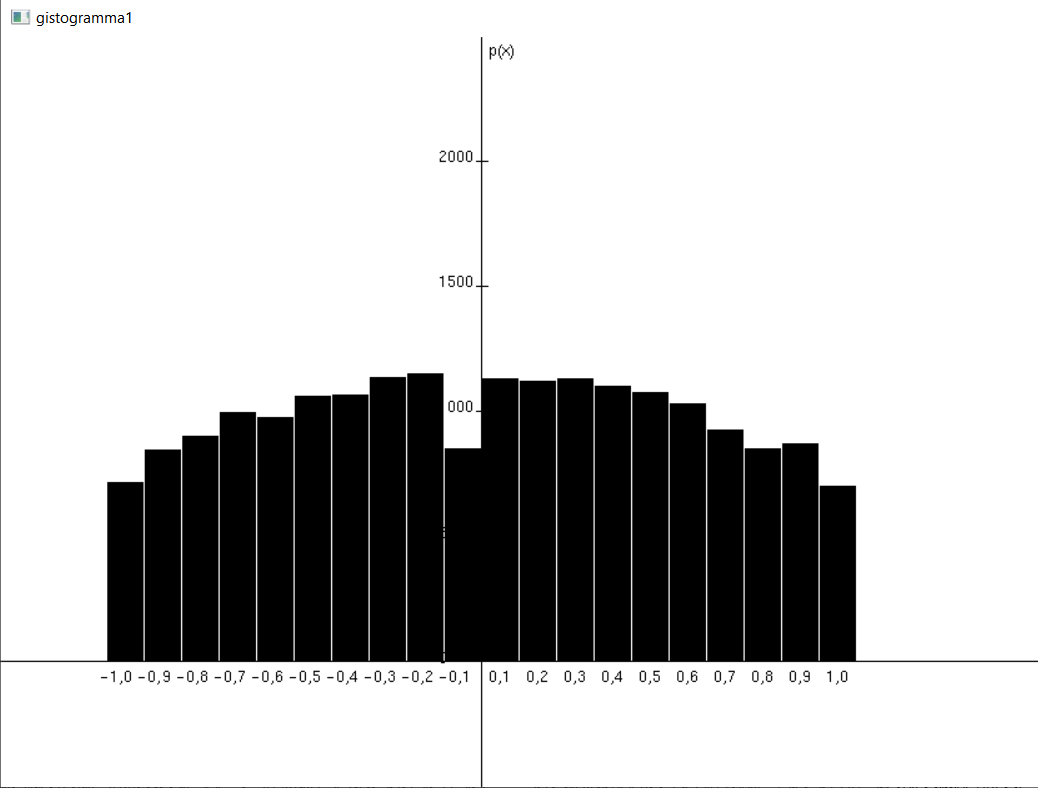
N = 15000

Программа выдает следующие графики:



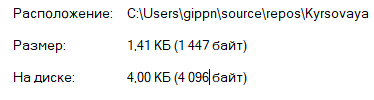


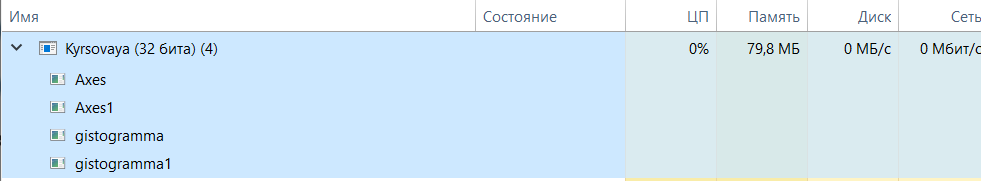




Для выполнения программы требуется 4 КБ (4096 байт)

долговременной памяти



79,8 МБ оперативной памяти

Среднее время работы программы измерено с помощью функции clock() и равняется 2.3мс.

# Описание команд и идентификаторов библиотеки glut.h

glClear(GL\_COLOR\_BUFFER\_BIT) – очищает указанный буфер;

GL\_COLOR\_BUFFER\_BIT – очистка буфера цвета;

glBegin(GL\_LINES); glEnd(); - Команды рисования вершин с заданием типа выводимого примитива

GL\_LINES – тип примитива (линия, строится по двум точкам)

glVertex(x,y) – определяет координаты вершины

glPopMatrix(); - Восстанавливает состояние матрицы из памяти (стека)

glPushMatrix(); - Сохраняет текущее состояние матрицы в памяти (стеке)

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15, strm[i]); - Выводит указанный символ на экран;

GLUT\_BITMAP\_9\_BY\_15 – задает тип шрифта

glRasterPos(x,y) – указывает расположение выводимого символа

glColor3f(r, g, b) – задает цвет примитива или символа в формате r,g,b;

glMatrixMode(GL\_PROJECTION); - выбирает активную матрицу из списка (стека)

(GL\_PROJECTION) – текущая матрица(проект)

Структура **BITMAPINFOHEADER** содержит информацию о размерах и цветовом формате

***biSize*** -Определяет число байтов необходимое для структуры.

***biWidth*** -Определяет ширину точечного рисунка, в пикселях.

***biPlanes*** - Устанавливает число плоскостей целевого устройства

glutInitDisplayMode(GLUT\_RGB); - Указывает цветовой режим

GLUT\_RGB – режим RGB

glutInitWindowSize(x, y); - инициализирует окно размерами x,y

glutCreateWindow(string) - задает название окна(string);

# Список используемой литературы

1. Ритчи Д. М., Керниган Б. У. «Язык программирования C»; «Вильямс» 2017г., 288с <http://www.opengl.org.ru/coding/glut/glut4.html> - описание некоторых функций библиотеки glut.
2. <http://www.codenet.ru/progr/opengl/opengl_03.php> - описание некоторых функций библиотеки glut.
3. [https://www.khronos.org/opencl/](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fwww.khronos.org%2Fopencl%2F) - официальный сайт GLUT
4. <http://grafika.me/node/137> - шрифты для вывода текста на экран