**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра САПР**

**ОТЧЁТ**

**По практической работе №3**

**По дисциплине «Программирование»**

Студент гр. 3351 \_\_\_\_\_\_\_\_\_\_\_\_ Морозов А.А.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_ Рыжов Н.Г.

Санкт-Петербург

2023

цель работы

Пользователь задает натуральные числа: **r** – радиус окружностей, **n** – исходное количество окружностей; **{X1,Y1}, …, {Xn, Yn}** – последовательность задаваемых генератором случайных числе координат точек на экране, представляющих собой центры окружностей радиуса **r**.

Проверить на примере следующего задания работу отладчика Си++:

1. Подсчитать количество непересекающихся окружностей.
2. Сформировать из исходной последовательности новую последовательность координат центров окружностей, которые не имеют общих точек с другими окружностями. Вывести результат на экран.
3. Нарисовать на экране непересекающиеся окружности, соединив их каждую с каждой линиями кратчайшей длины. Линии должны быть нарисованы от края окружности, а не от центра. Если линия проходит через другую окружность, то ее участок, проходящий внутри окружности, не отображается.
4. В разделе Руководство пользователя помимо окна результатов приведите скриншоты окна программы с установленными точками останова, а также окна Контрольные значения со значениями произвольно выбранных переменных.

Требование: для задания исходной и новой последовательности координат точек сформировать массивы. На примере этой программы изучите различные режимы и команды отладчика.

**ТЕКСТ ПРОГРАММЫ “Непересекающиеся окружности”**

#include <SFML/Graphics.hpp>

#include <iostream>

#include <conio.h>

#include <windows.h>

#include <cmath>

const int window\_height = 1000, window\_width = 1000; // Параметры создаваемого окна

const float pi = 3.14159265357989; // Число ПИ

using namespace std;

int main() {

int i, k, circle\_num, circle\_radius, correctnumcords = 0; // Инициализация всех необходимых переменных

bool Closed = false, Correct = true;

float sin, cos;

unsigned int seed = time(nullptr); // Для генератора псевдослучайных чисел

srand(seed);

setlocale(LC\_ALL, "RU"); // Локализация на русский язык

while (true) { // Ввод радиуса с проверкой на верность

cout << "Введите радиус окружностей: " << endl;

cin >> circle\_radius;

if (circle\_radius > 0) break;

cout << "Ошибка: Введён нулевой радиус окружностей. Повторите ввод: " << endl;

}

while (true) { // Ввод количества окружностей с проверкой правильности введенных значений

cout << "Введите количество окружностей: " << endl;

cin >> circle\_num;

if (circle\_num > 0) break;

cout << "Ошибка: Введено нулевое количество окружностей. Повторите ввод: " << endl;

}

int\*\* array\_nc{ new int\* [circle\_num] }, \*\* correctmatrix{ new int\* [circle\_num] }; // Массивы для всех сгенерированных значений и правильных значений

for (k = 0; k < circle\_num; k++) { // Генерация значений координат центров окружностей

array\_nc[k] = new int[2] { (rand() % window\_width), (rand() % window\_height) };

}

for (k = 0; k < circle\_num; k++) { // Проверка сгенерированных значений на пересечения

Correct = true;

for (i = 0; i < circle\_num; i++) {

if (k != i) {

if (sqrt(pow(array\_nc[k][0] - array\_nc[i][0], 2) + pow(array\_nc[k][1] - array\_nc[i][1],

2)) <= 2 \* circle\_radius) { // Поиск пересекающихся окружностей

Correct = false;

break;

}

}

}

if (Correct) { // Добавление верных значений в массив с корректными координатами, увеличение счетчика - количества верных координат

correctmatrix[correctnumcords] = new int[2] { array\_nc[k][0], array\_nc[k][1] };

correctnumcords++;

}

}

cout << "Координаты непересекающихся окружностей:" << endl; // Вывод сообщения на консоль

cout << "X" << "\t" << "Y" << endl;

for (k = 0; k < correctnumcords; k++) {

cout << correctmatrix[k][0] << "\t" << correctmatrix[k][1] << endl; // Вывод корректных координат на консоль

}

sf::RenderWindow window(sf::VideoMode(window\_width, window\_height), L"Окружности"); // Создание окна

window.clear();

while (window.isOpen()) {

sf::Event event{};

while (window.pollEvent(event)) { // Отслеживание закрытия

if (event.type == sf::Event::Closed) {

window.close();

Closed = true;

break;

}

}

if (Closed) break;

for (k = 0; k < correctnumcords; k++) { // Рисование и создание линий

for (i = 0; i < correctnumcords; i++) {

sin = abs(correctmatrix[k][1] - correctmatrix[i][1]) / sqrt(pow(correctmatrix[k][0] - correctmatrix[i][0], 2) +

pow(correctmatrix[k][1] - correctmatrix[i][1], 2)); // Ищу синус

cos = abs(correctmatrix[k][0] - correctmatrix[i][0]) / sqrt(pow(correctmatrix[k][0] - correctmatrix[i][0], 2) +

pow(correctmatrix[k][1] - correctmatrix[i][1], 2)); // Ищу косинус

sf::VertexArray line(sf::Lines, 2); // Строю линию

line[0].color = sf::Color::White;

line[1].color = sf::Color::White;

if (correctmatrix[k][0] < correctmatrix[i][0] and correctmatrix[k][1] < correctmatrix[i][1]) {

line[0].position = sf::Vector2f(correctmatrix[k][0] + circle\_radius \* cos, correctmatrix[k][1] +

circle\_radius \* sin);

line[1].position = sf::Vector2f(correctmatrix[i][0] - circle\_radius \* cos, correctmatrix[i][1] -

circle\_radius \* sin);

}

else if (correctmatrix[k][0] < correctmatrix[i][0] and correctmatrix[k][1] > correctmatrix[i][1]) {

line[0].position = sf::Vector2f(correctmatrix[k][0] + circle\_radius \* cos, correctmatrix[k][1] -

circle\_radius \* sin);

line[1].position = sf::Vector2f(correctmatrix[i][0] - circle\_radius \* cos, correctmatrix[i][1] +

circle\_radius \* sin);

}

else if (correctmatrix[k][0] > correctmatrix[i][0] and correctmatrix[k][1] < correctmatrix[i][1]) {

line[0].position = sf::Vector2f(correctmatrix[k][0] - circle\_radius \* cos, correctmatrix[k][1] +

circle\_radius \* sin);

line[1].position = sf::Vector2f(correctmatrix[i][0] + circle\_radius \* cos, correctmatrix[i][1] -

circle\_radius \* sin);

}

else {

line[0].position = sf::Vector2f(correctmatrix[k][0] - circle\_radius \* cos, correctmatrix[k][1] -

circle\_radius \* sin);

line[1].position = sf::Vector2f(correctmatrix[i][0] + circle\_radius \* cos, correctmatrix[i][1] +

circle\_radius \* sin);

}

window.draw(line);

}

}

for (k = 0; k < correctnumcords; k++) { // Создание и рисование окружностей

sf::CircleShape shape(circle\_radius);

shape.setPosition(correctmatrix[k][0] - circle\_radius, correctmatrix[k][1] - circle\_radius);

window.draw(shape);

}

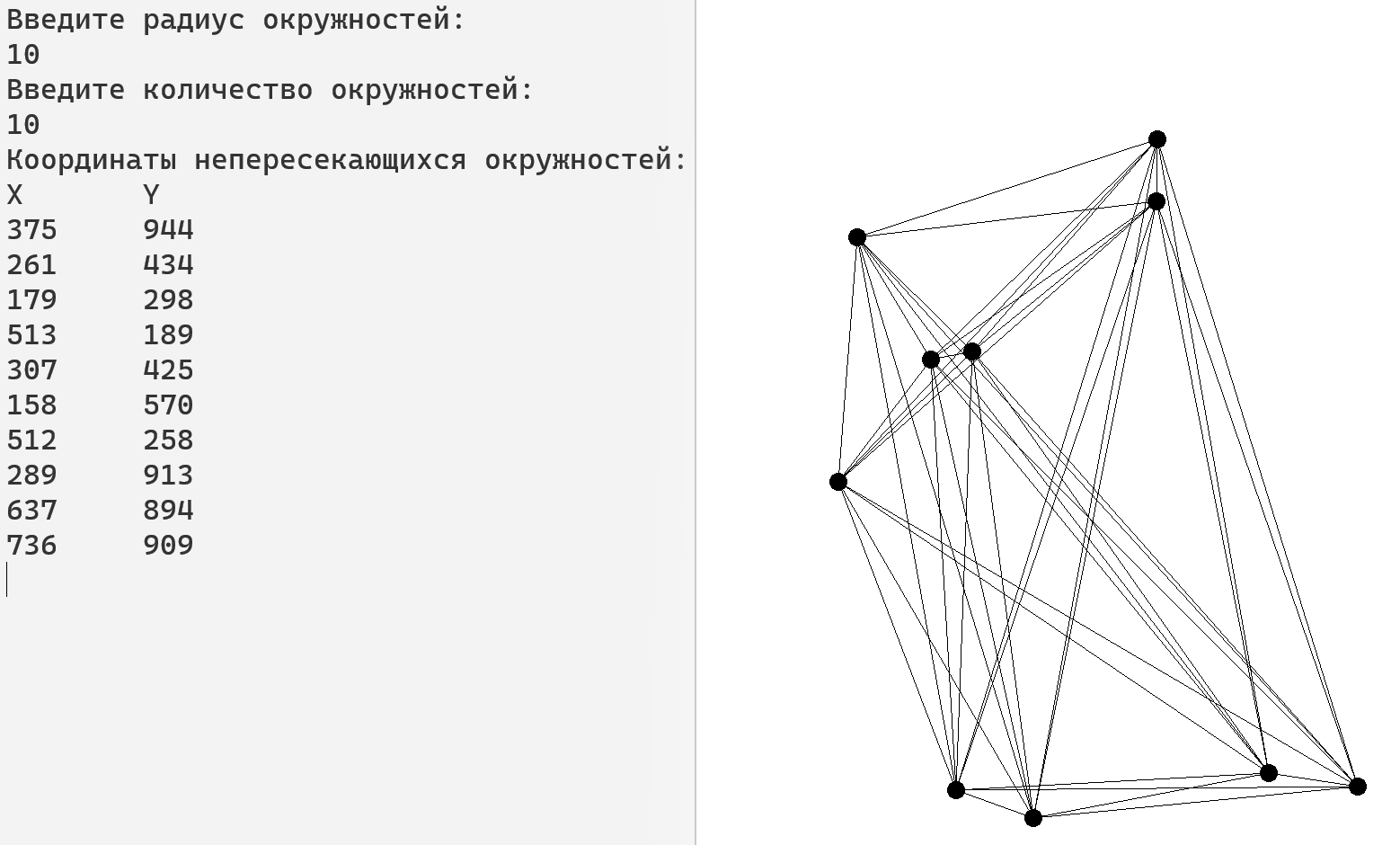
window.display(); // Отображение фигур из буфера

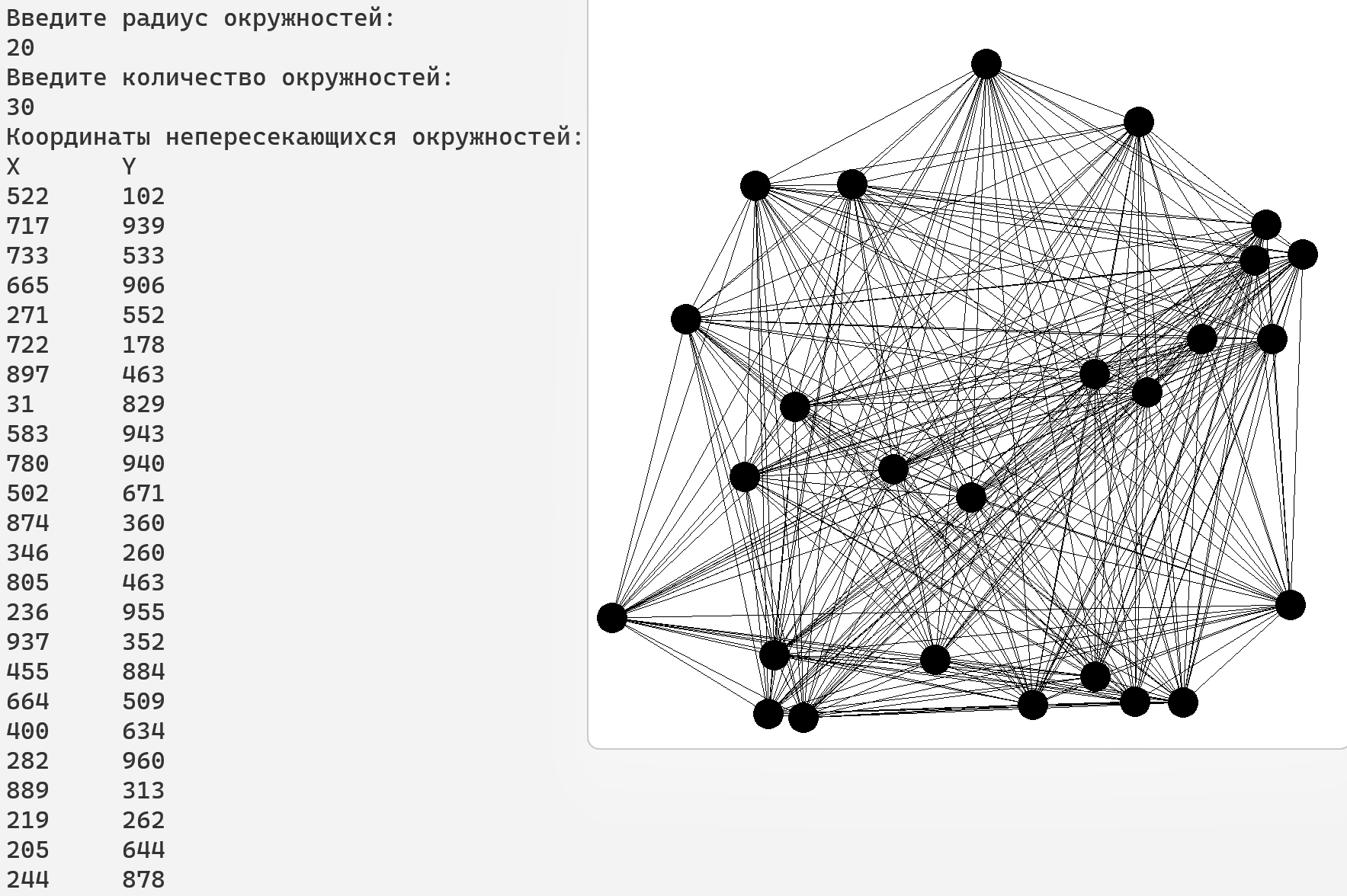
}

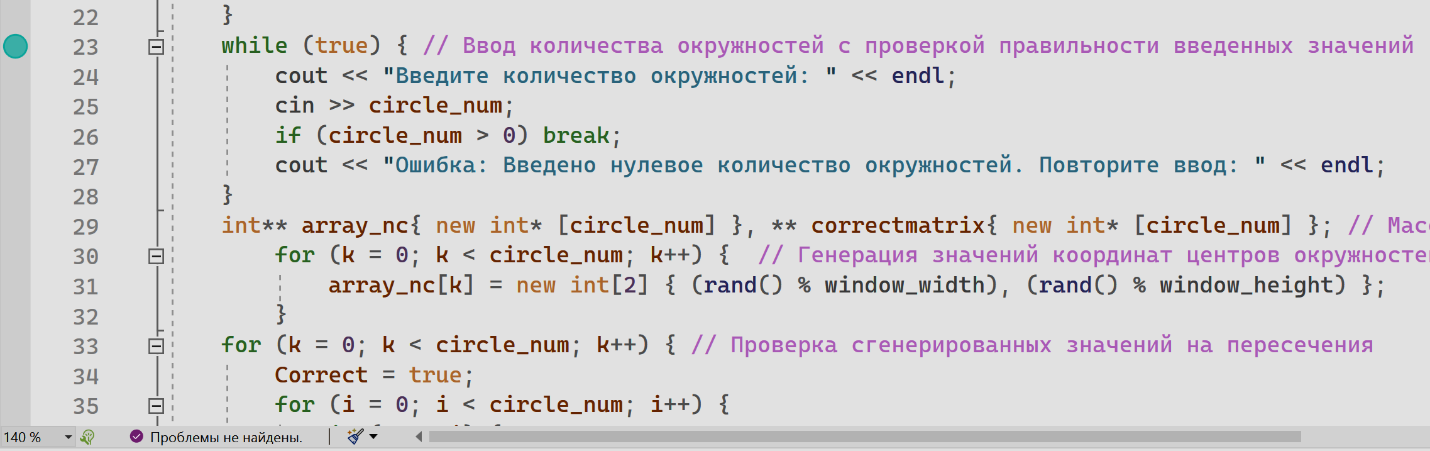
return 0;

}

**Результаты выполнения программы**







**Пути дальнейшего улучшения программы**

1. Улучшение визуальной части программы
2. Смена вида вывода значений в консоли