МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ М.В. ЛОМОНОСОВА

ФИЛИАЛ МОСКОВСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА ИМЕНИ М.В. ЛОМОНОСОВА

ФАКУЛЬТЕТ «ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ»

Направление подготовки «Прикладная математика и информатика»

010400.62 (бакалавр)

ПАНЧЕНКО ВЛАДИСЛАВ АНДРЕЕВИЧ

студент учебной группы ПМ-401

«СИМВОЛИЧЕСКИЙ ОБРАЗ ОТОБРАЖЕНИЯ ЖЮЛИА И ЛОКАЗИЗАЦИЯ ЦЕПНО-РЕКУРРЕНТНОГО МНОЖЕСТВА»

Отчёт

Проверил доктор физико-математических наук,

профессор кафедры прикладной математики

Осипенко Георгий Сергеевич

Севастополь, 2021

В ходе проделанной работыбыла поставлена задача изучения построения цепно-рекуррентного множества символических образов.

Для такого построения было необходимо решить задачу поиска сильно связных компонент в графе.

Для проверки работы алгоритма было взято отображение Жюлиа:

**Компонентой сильной связности** называется такое (максимальное по включению) подмножество вершин, что любые две вершины этого подмножества достижимы друг из друга, т.е. для :

**Областью сильной связности** называется множество вершин компоненты сильной связности.

Для нахождения таких компонент был использован алгоритм Косараю, который работает за линейное время благодаря двойному поиску в глубину: одному – в графе и второму в графе , где - состоит из вершин и ребер, направление которых было изменено на обратное, графа .

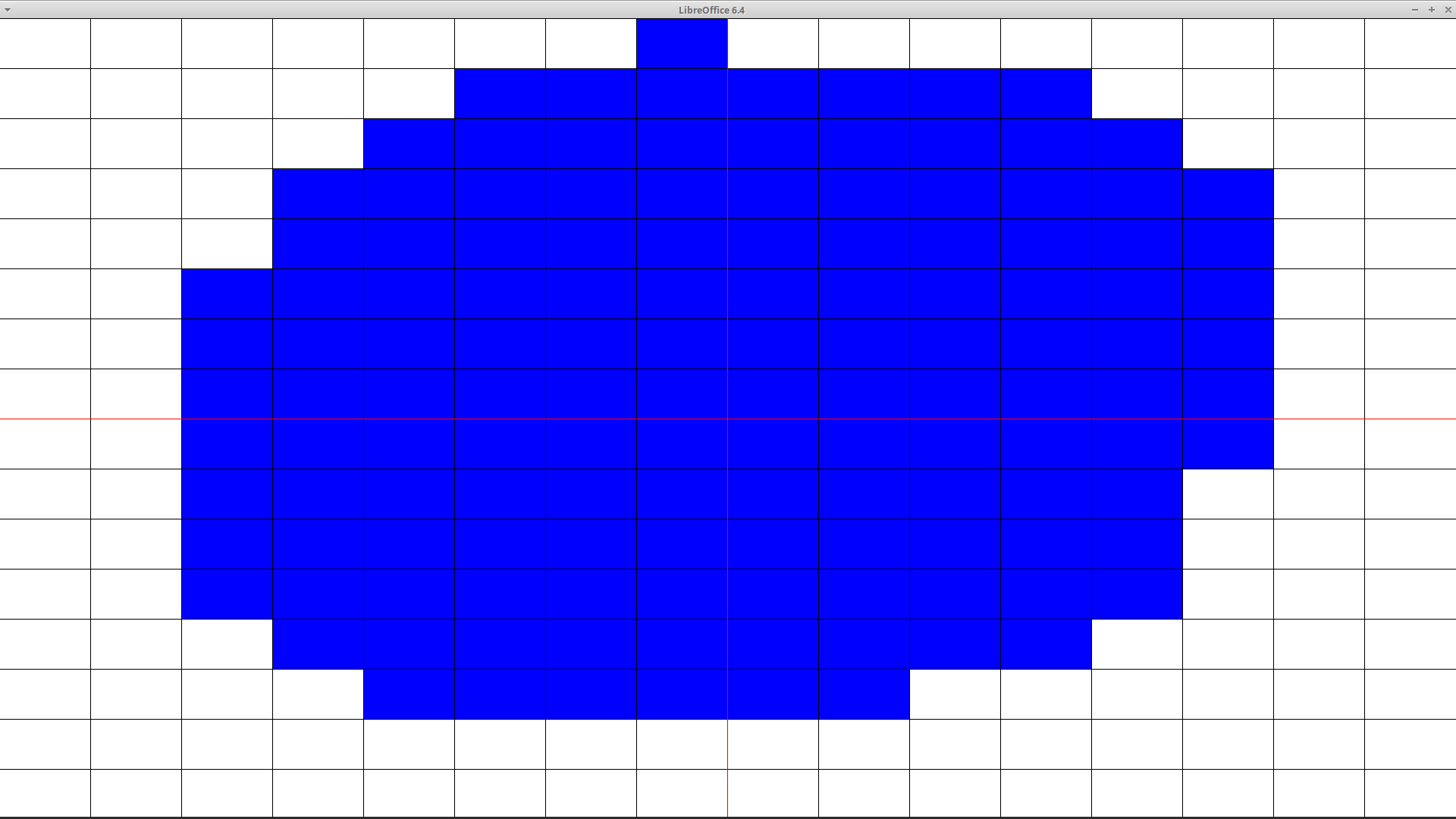
**Алгоритм локализации цепно-рекуррентного множества состоит в следующем:**

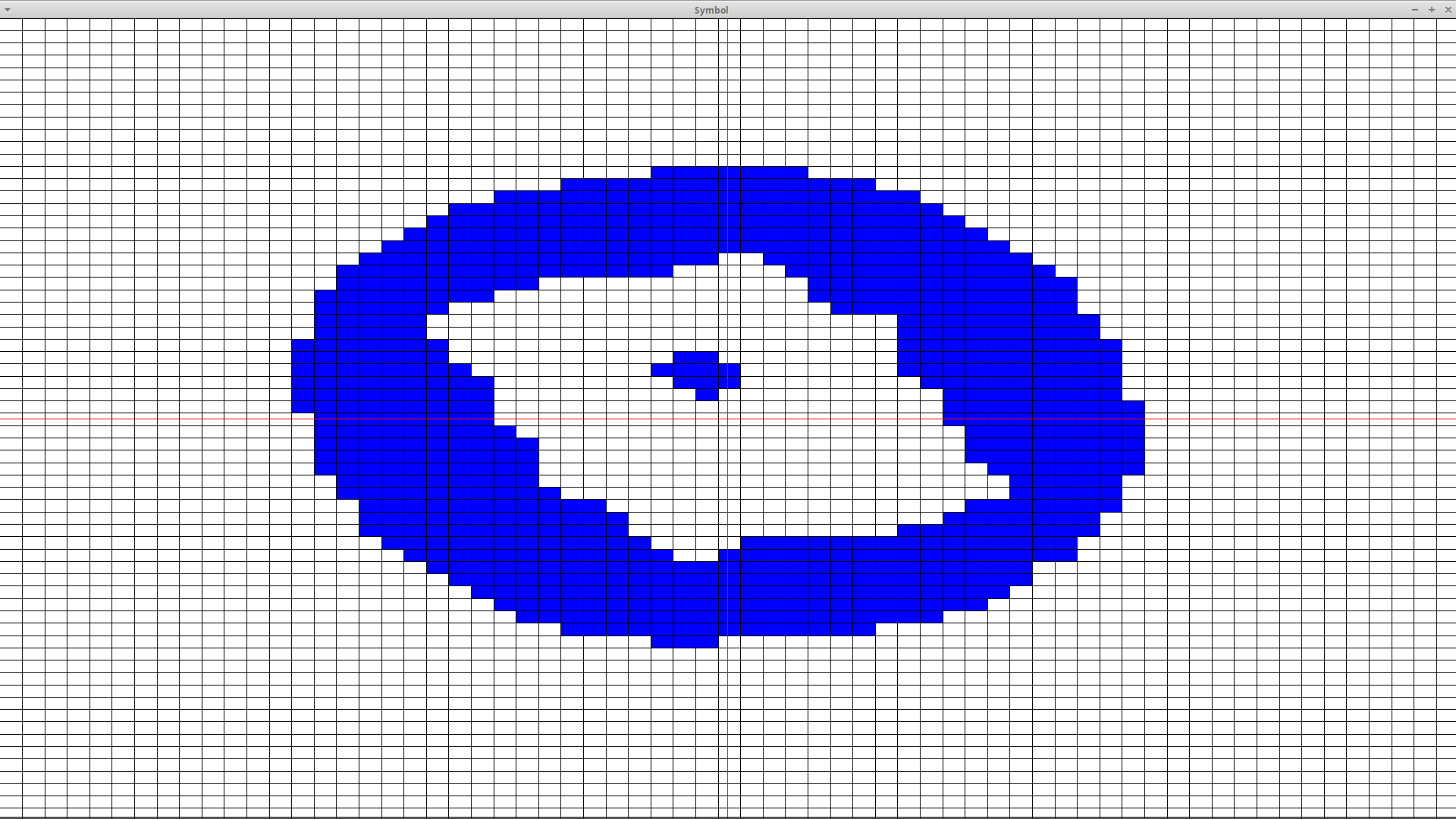
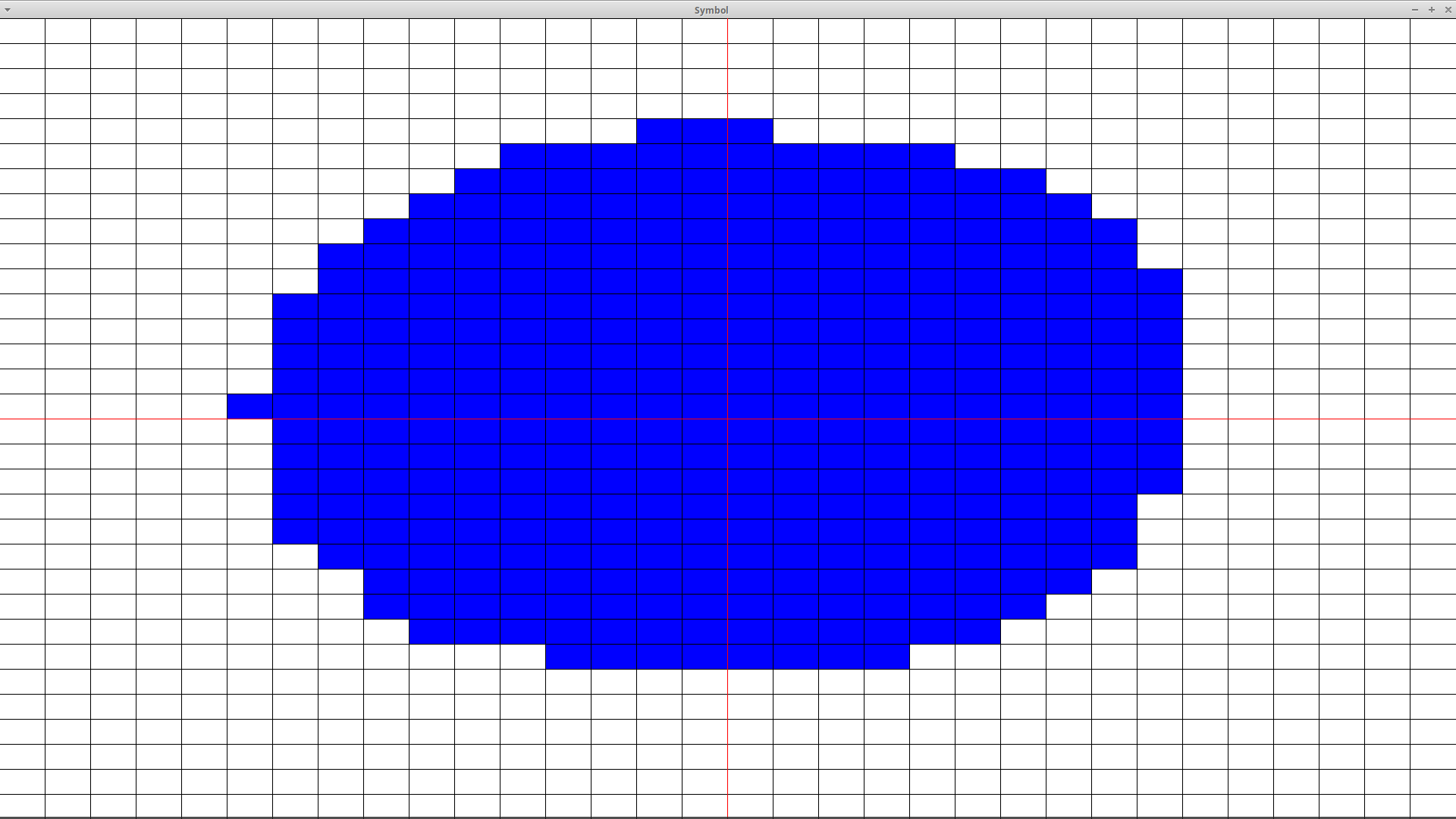
1. Строим исходное покрытие C компакта D. Находим символический образ G отображения f. Заметим, что ячейки исходного покрытия могут иметь произвольный диаметр d0.
2. Выделяем на графе G возвратные вершины {𝑖𝑘}. Если множество таких вершин пустое, значит локализуемое цепно-рекуррентное множество является пустым и процесс его локализации прекращается. Иначе,
3. используя их, находим замкнутую окрестность P = {x ∈ 𝐷𝑖𝑘 : 𝑖𝑘 — возвратные} цепно-рекуррентного множества Q.
4. Разбиваем ячейки {𝐷𝑖𝑘}, соответствующие возвратным вершинам символического образа и, таким образом, определяем новое покрытие.
5. Строим символический образ G для нового покрытия.
6. Переходим ко второму пункту, если размеры ячеек построенного символического образа достаточно велики. Повторяя процесс последовательного измельчения покрытия, мы получаем последовательность окрестностей 𝑃1,𝑃2,𝑃3. . . цепно-рекуррентного множества 𝑄 и последовательность наибольших диаметров 𝑑1,𝑑2,𝑑3, . . . ячеек, соответствующих возвратным вершинам символического образа для покрытия 𝐶𝑘

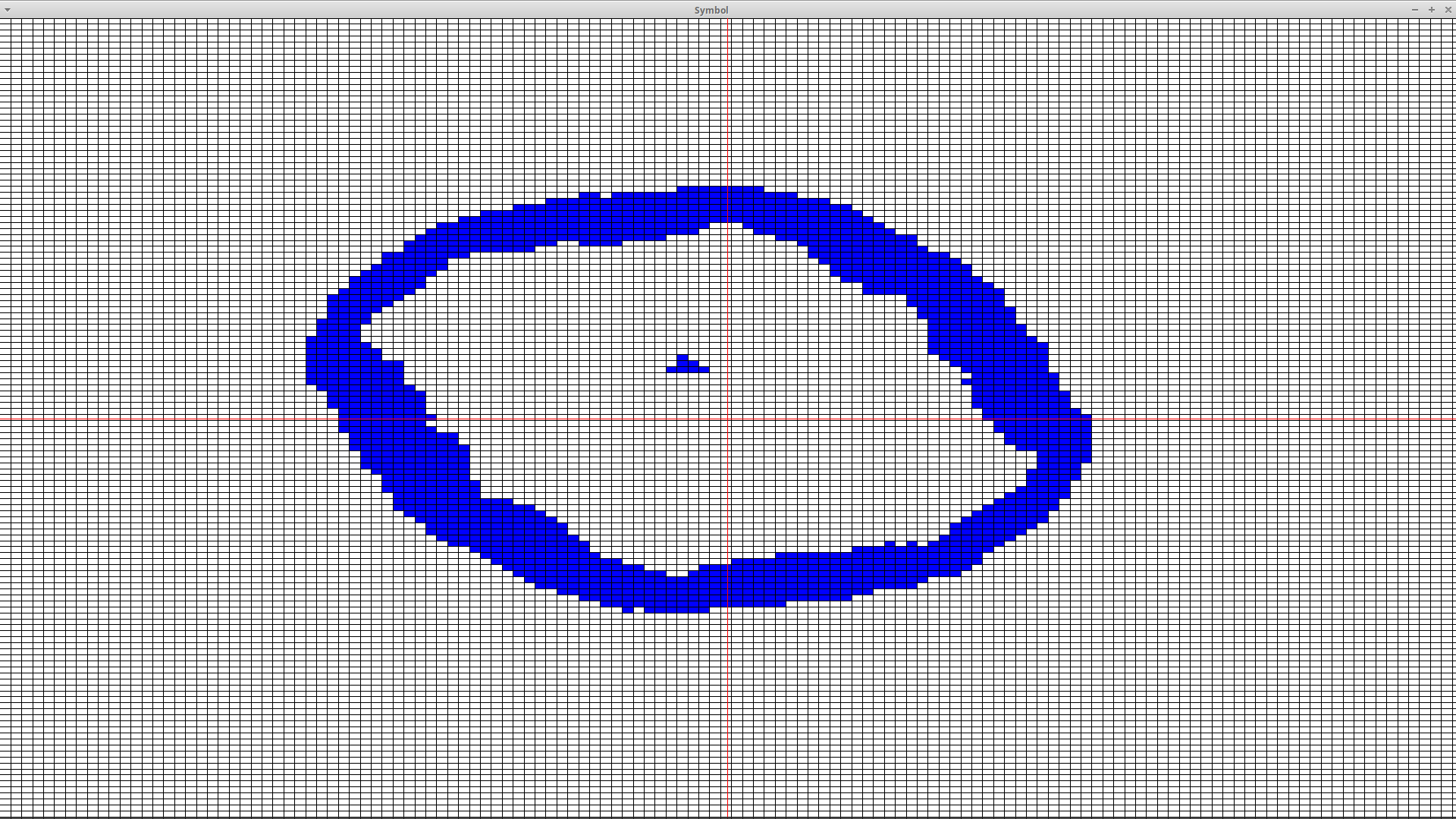
Реализация написана самостоятельно, алгоритм для поиска компонент сильной связности был использован алгоритмической библиотеки языка C++

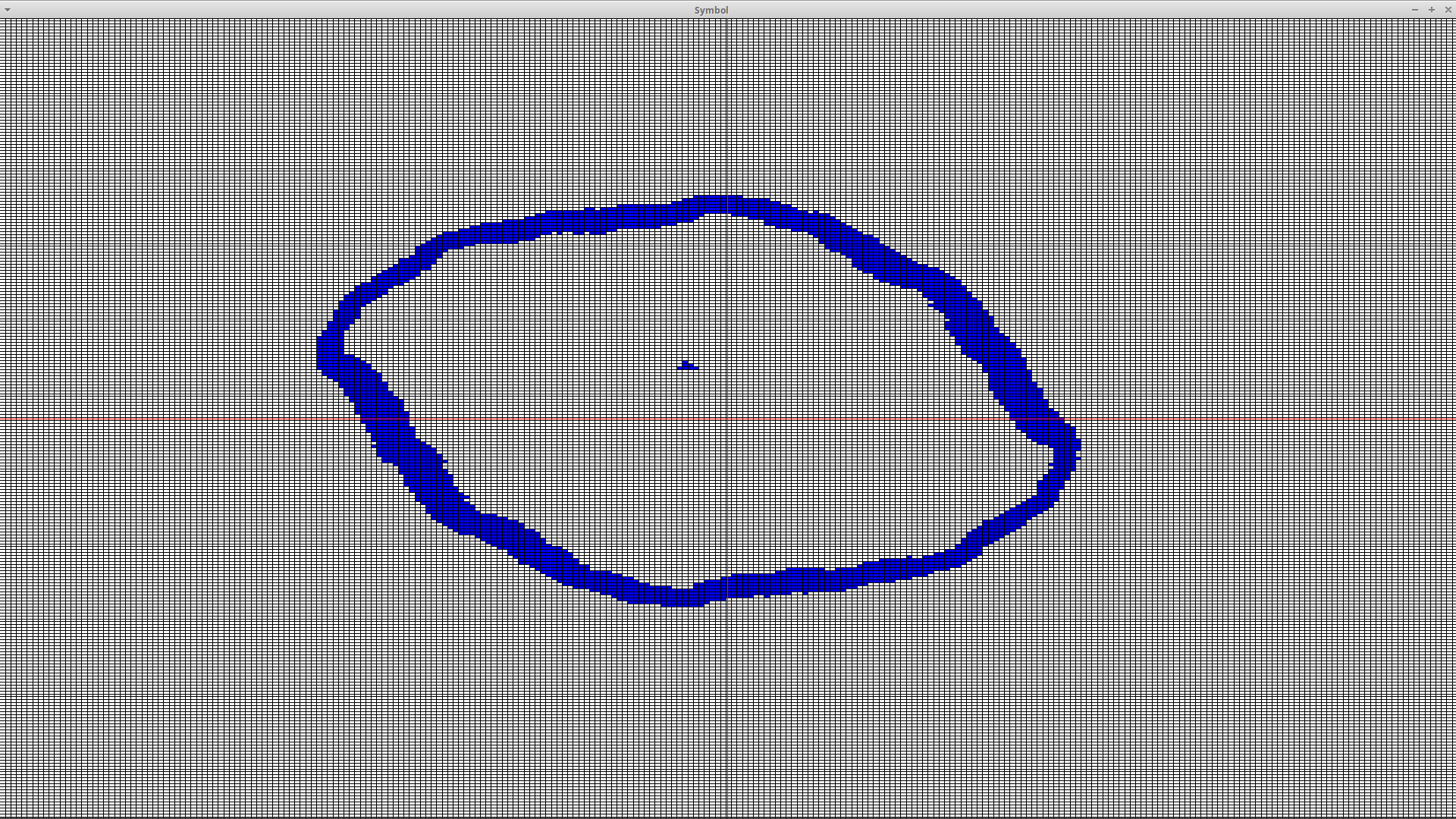
Для данной программы использовался язык программирования С++ на операционной системе Linux, дистрибутив Xubuntu 20.04.3 LTS (Focal Fossa)

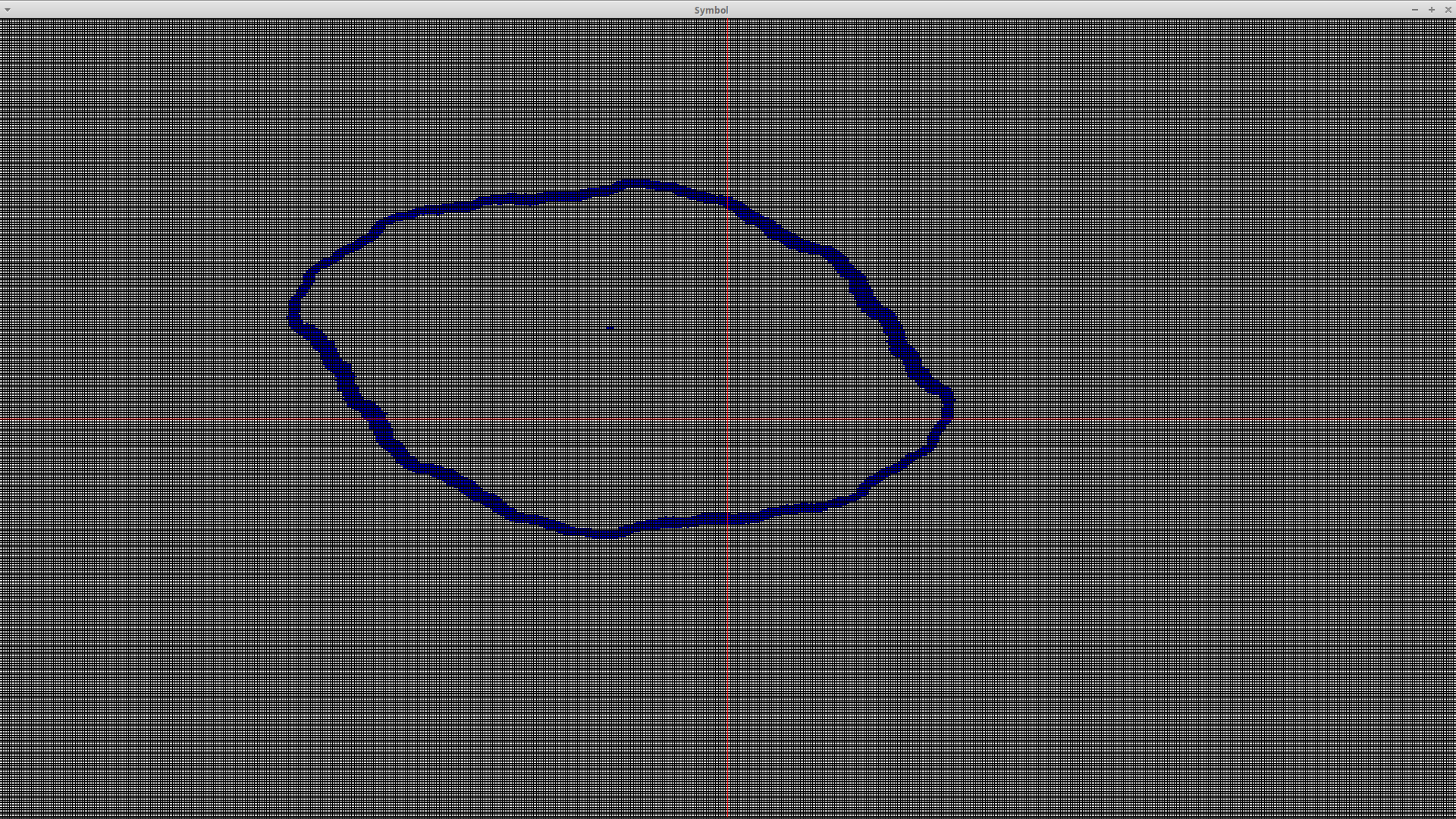
Ниже приведены результаты для 7 последовательных итераций.

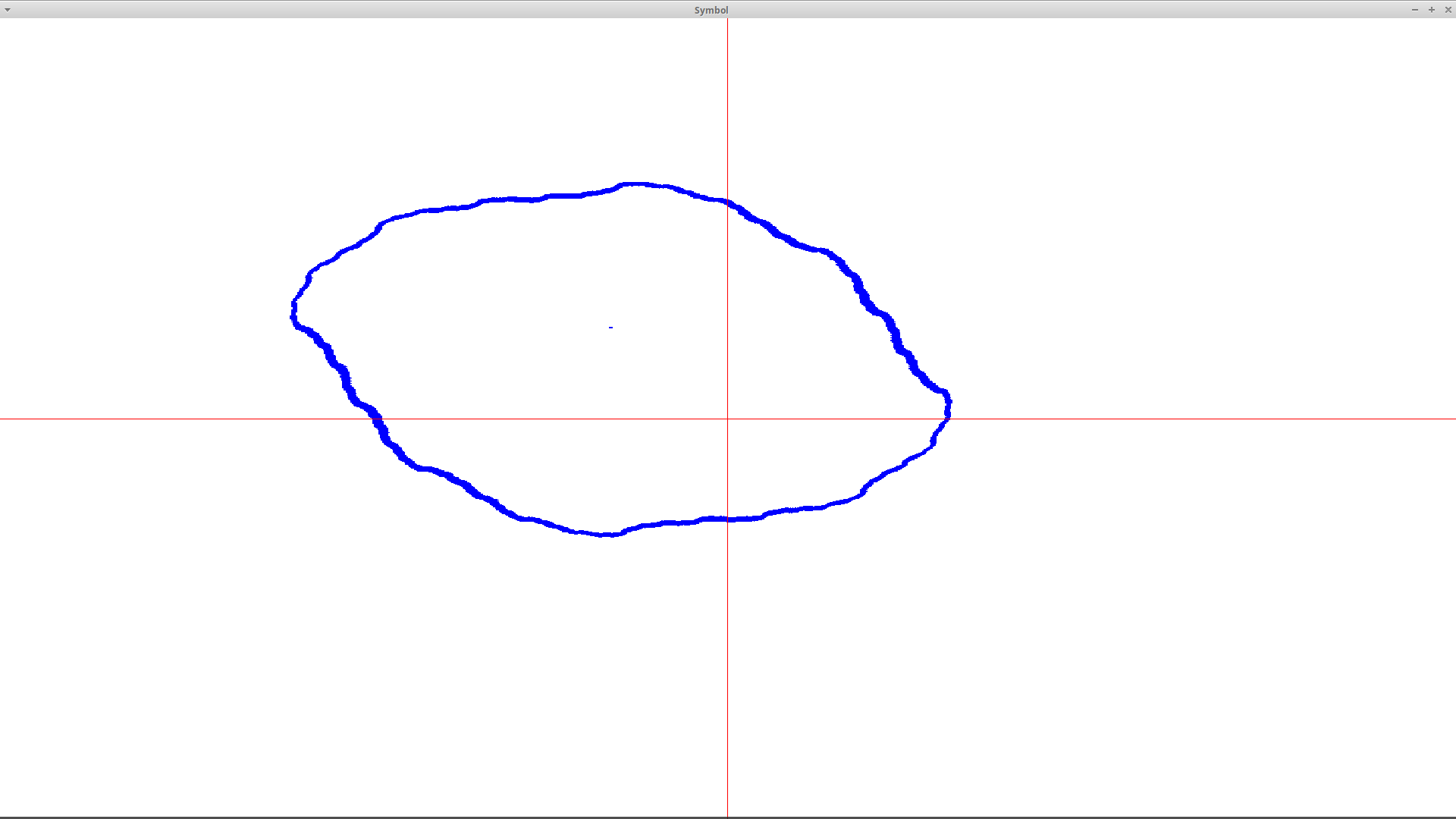












Время выполнения на конфигурации

MemTotal: 8051816 kB

Intel(R) Core(TM) i5-2450M CPU @ 2.50GHz

GeForce 610M

по итерациям:

1. 0,002963
2. 0,016224
3. 0,043329
4. 0,146370
5. 0,555902
6. 2,206925
7. 8,821445