Федеральное государственное бюджетное образовательное учреждение
высшего образования «Московский государственный технический университет
имени Н. Э. Баумана» (национальный исследовательский университет)

Дисциплина: «Анализ алгоритмов»
Отчет по рубежному контролю №1

Тема работы:
«Дерево Пифагора»

Студент: Левушкин И. К.
Группа: ИУ7-52Б
Преподаватели: Волкова Л. Л.,
Строганов Ю. В.

Москва, 2019 г.

Содержание

Введение

Цель лабораторной работы: При помощи конечных автоматов и регулярных выражений написать программу, находящую группы факультетов ИУ, ИБМ и Э в тексте.

Задачи работы:

1) изучить работу регулярных выражений и конечных автоматов;
2) создать конечный автомат;
3) реализовать поставленную задачу с использованием конечного автомата;
4) реализовать поставленную задачу с использованием регулярные выражения;

5) сравнить время выполнения программы, использующую конечный автомат, и программу, использующую регулярные выражения;

6) описать и обосновать полученные результаты в отчете о рубежном контроле работе, выполненного как расчётно-пояснительная записка.

1 Аналитический раздел

Дерево Пифагора — разновидность фрактала, основанная на фигуре, известной как «Пифагоровы штаны».

1.1 История

Пифагор, доказывая свою знаменитую теорему, построил фигуру, где на сторонах прямоугольного треугольника расположены квадраты. В наш век эта фигура Пифагора выросла в целое дерево. Впервые дерево Пифагора построил А. Е. Босман (1891—1961) во время второй мировой войны, используя обычную чертёжную линейку.

1.2 Особенности

Одним из свойств дерева Пифагора является то, что если площадь первого квадрата равна единице, то на каждом уровне сумма площадей квадратов тоже будет равна единице.

Если в классическом дереве Пифагора угол равен 45 градусам, то также можно построить и обобщённое дерево Пифагора при использовании других углов. Такое дерево часто называют обдуваемое ветром дерево Пифагора. Если изображать только отрезки, соединяющие каким-либо образом выбранные «центры» треугольников, то получается обнаженное дерево Пифагора [4] .

1.3 Примеры

На рисунках 1-3 представлены примеры пифагора дерева при различных параметрах.

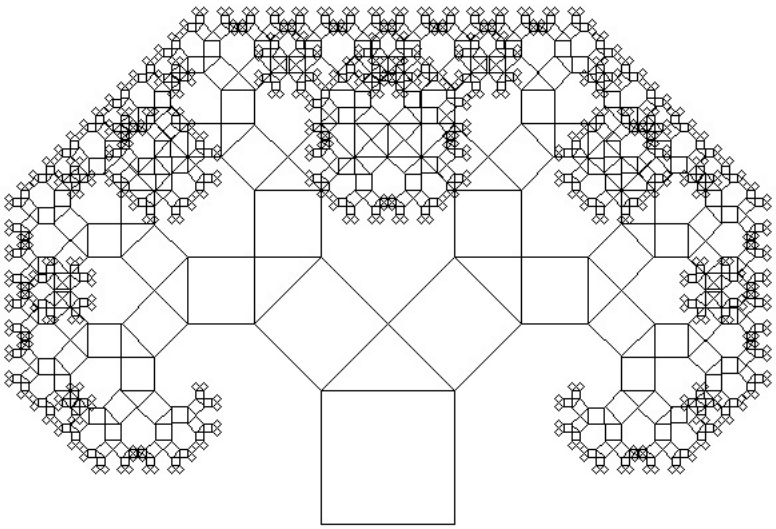

Рис. 1: Классическое дерево Пифагора

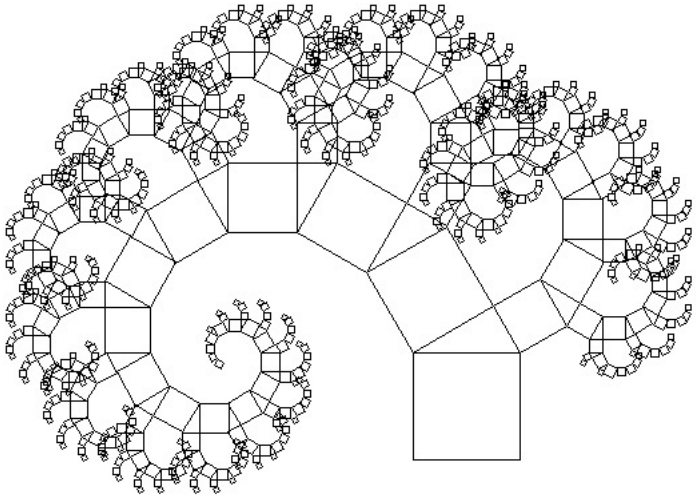

Рис. 2: Обдуваемое дерево Пифагора

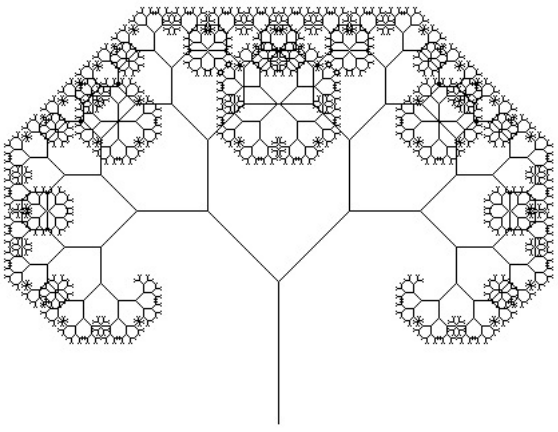

Рис. 3: Обнаженное дерево Пифагора

1.4 Выводы

В рамках данного рубежного контроля было решено реализовать обнаженное дерево Пифагора в программе для курсового проекта по Компьютерной графике, так как она наилучшим образом подходит для реализации данного алгоритма.

2 Конструкторский раздел

Ниже представлен алгоритм построения дерева Пифагора:

1. Строим вертикальный отрезок
2. Из верхнего конца этого отрезка рекурсивно строим еще 2 отрезка под определенными углами
3. Вызываем функцию построения двух последующих отрезков для каждой ветви дерева

3 Технологический раздел

Здесь описываются требования к программному обеспечению и средства реализации, приводятся листинги программы и тестовые данные.

3.1 Требования к программному обеспечению

Входные данные:

- угол между ветками;
- толщина ветки;
- длина ветки.

Выходные данные: Массив точек, связанных между собой отрезками.

3.2 Средства реализации

Для реализации поставленной задачи был использован язык программирования C++ [1]. Проет был выполнен в среде QT Creator [2]. Для измерения процессорного времени была использована ассемблерная инструкция rdtsc [3].

3.3 Листинг программы

Реализованный алгоритм представлен в листинге 1.

Листинг 1: Реализация обнаженного дерева Пифагора

```
1     void model_nodes_fill(std::shared_ptr<Model> model, Binary_Tree *points)
2     {
3     Node node1(points->br.point1.x, points->br.point1.y, points->br.point1.z
          , {0, 0, 0});
4     Node node2(points->br.point2.x, points->br.point2.y, points->br.point2.z
          , {0, 0, 0});
5
6     model->addNode(node1);
7     model->addNode(node2);
8
9     if (points->left != nullptr)
10    {
11    model_nodes_fill(model, points->left);
12    }
13
14    if (points->right != nullptr)
15    {
16    model_nodes_fill(model, points->right);
17    }
18    }
19
20
21    double calc_a_r(double a, double alpha)
22    {
23    return a * (sin(alpha) + cos(alpha) / 2);
24    }
25
26    double calc_pifagor_coord(double x, double a_r, double cos_alpha)
27    {
28    return x + a_r * cos_alpha;
29    }
30
31    void pifagor_algorithm_iter(Branch &br, Branch &next_br, double
          next_angle, double a, double alpha)
32    {
33    next_br.point1 = br.point2;
34    double a_r = calc_a_r(a, alpha);
35
36    double x = calc_pifagor_coord(next_br.point1.x, a_r, cos(next_angle));
37    double y = calc_pifagor_coord(next_br.point1.y, a_r, sin(next_angle));
38    next_br.point2.x = x;
39    next_br.point2.y = y;
40    next_br.point2.z = 0;
41    }
42
43
44    void create_next_pifagor_branch(Binary_Tree *tree, vector<double> &
          branch_radius, double next_angle, double alpha, size_t max_iteration,
           double a)
45    {
```

```
46      if (tree->br.depth == max_iteration)
47      {
48      return;
49      }
50
51      //double a = calc_length(tree->br.point1, tree->br.point2);
52
53      tree->left = new Binary_Tree();
54      tree->left->left = nullptr;
55      tree->left->right = nullptr;
56
57      tree->left->br.depth = tree->br.depth + 1;
58      pifagor_algorithm_iter(tree->br, tree->left->br, M_PI / 2 + next_angle,
           a, M_PI / 2 - alpha);
59      tree->left->br.radius = tree->br.radius / 1.1;
60      branch_radius.push_back(tree->left->br.radius);
61
62      create_next_pifagor_branch(tree->left, branch_radius, alpha + next_angle
           , alpha, max_iteration, cos(alpha) * a);
63
64      tree->right = new Binary_Tree();
65      tree->right->left = nullptr;
66      tree->right->right = nullptr;
67
68      tree->right->br.depth = tree->br.depth + 1;
69      pifagor_algorithm_iter(tree->br, tree->right->br, next_angle, a, alpha);
70      tree->right->br.radius = tree->br.radius / 1.1;
71      branch_radius.push_back(tree->right->br.radius);
72      create_next_pifagor_branch(tree->right, branch_radius, next_angle - (
           M_PI / 2 - alpha), alpha, max_iteration, sin(alpha) * a);
73
74      }
75
76      void pifagor_algorithm(Binary_Tree *tree, vector<double> &branch_radius,
            double a, double alpha, size_t iteration)
77      {
78      tree->br.depth = 0;
79      tree->br.point1 = {0, 0, 0};
80      tree->br.point2 = {0, a, 0};
81      tree->br.radius = branch_radius[0];
82      create_next_pifagor_branch(tree, branch_radius, alpha, alpha, iteration,
            a);
83      }
84
85      std::shared_ptr<Transformed_Model> Pifagor::buildModel(tree_params
           params, std::shared_ptr<Transform_Tree> tr_tree)
86      {
87      model = std::make_shared<Model>();
88
89      double a = params.body.length;
90      double alpha = params.angles.alpha1;
91
92      size_t iteration = 11;
93
94      size_t branches = get_tree_branches(iteration);
```

```
95
96      size_t nodes_amount = branches * 2;
97
98      vector<double> branch_radius;
99
100     branch_radius.push_back(params.body.radius);
101
102     Binary_Tree *nodes = new Binary_Tree();
103     nodes->left = nullptr;
104     nodes->right = nullptr;
105
106     pifagor_algorithm(nodes, branch_radius, a, alpha, iteration);
107
108     model_nodes_fill(this->model, nodes);
109
110     edges_fill(this->model, nodes_amount, branch_radius);
111
112     this->model->setAccuracy(params.body.accuracy);
113     return tr_tree.get()->buildModel(model, params.tree_clr.clr);
114     }
```

4 Исследовательский раздел

В разделе представлены примеры выполнения программы.

4.1 Примеры работы

На рис. 4-8 приведены примеры работы программы.

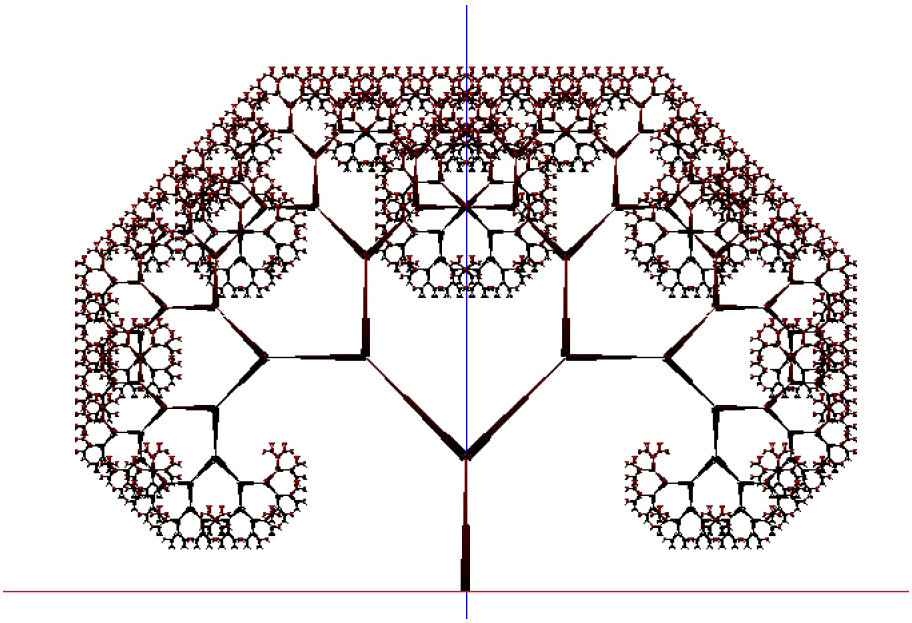

Рис. 4: Обнаженное классическое дерево Пифагора (Угол 45 градусов)

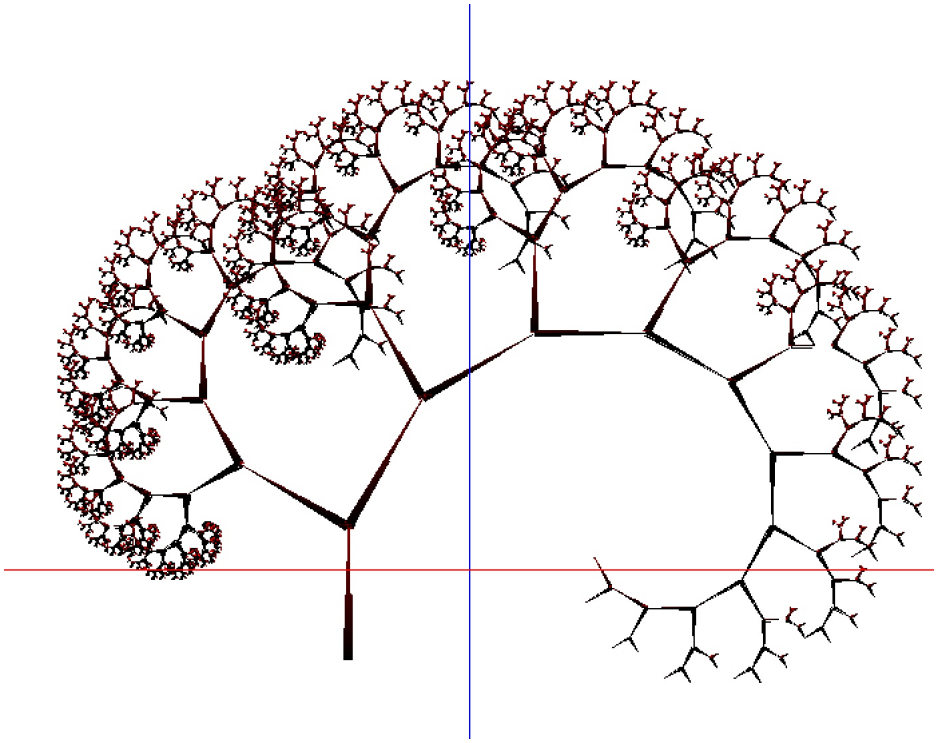

Рис. 5: Обнаженное обдуваемое дерево Пифагора (Угол 30 градусов)

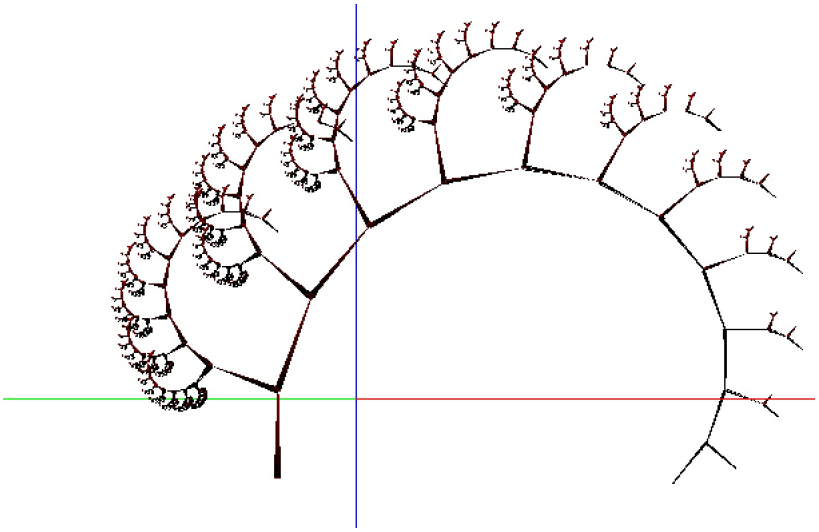

Рис. 6: Обнаженное обдуваемое дерево Пифагора (Угол 20 градусов)

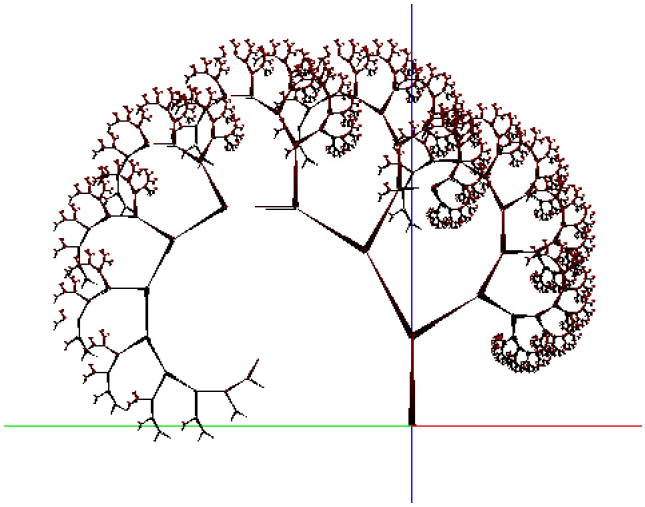

Рис. 7: Обнаженное обдуваемое дерево Пифагора (Угол 60 градусов)

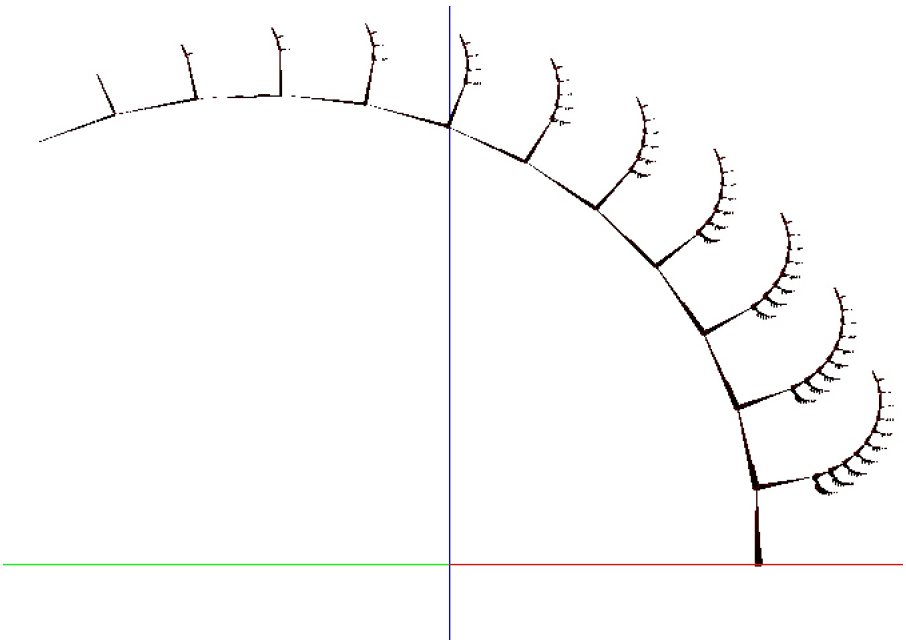

Рис. 8: Обнаженное обдуваемое дерево Пифагора (Угол 10 градусов)

Заключение

В ходе выполнения данной лабораторной работы был изучен алгоритм построения фрактала дерева Пифагора. Алгоритм был разработан и реализацован.

Список литературы

[1] ISO/IEC JTC1 SC22 WG21 N 3690 «Programming Languages — C++» [Электронный ресурс]. – Режим доступа: https://devdocs.io/cpp/, свободный. (Дата обращения: 29.09.2019 г.)

[2] QT Creator Manual [Электронный ресурс]. – Режим доступа: https://doc.qt.io/qtcreator/index.html, свободный. (Дата обращения: 29.09.2019 г.)

[3] Microsoft «rdtsc» [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/cpp/intrinsics/rdtsc?view=vs-2019, свободный. (Дата обращения: 29.09.2019 г.)

[4] Pifagor algorithm [Электронный ресурс]. - Режим доступа: http://grafika.me/node/87 - (28.11.2019)