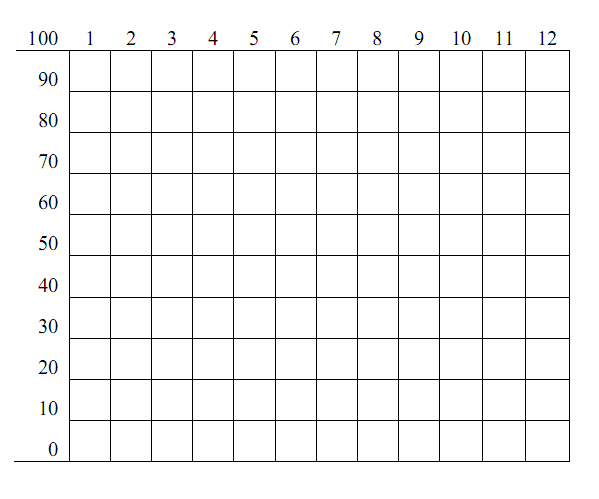
**Министерство образования и науки РФ**

**ФГБОУ ВПО «Уфимский государственный авиационный технический университет»**

**Кафедра** Высокопроизводительных вычислительных технологий и систем

****

**Реализация алгоритма поиска расстояний и пути от каждой вершины до остальных путем удалении n-2 вершин и добавлении их обратно**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе по дисциплине**

**«Операционные сети и Системы 2»**

|  |
| --- |
| ... ПЗ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа МКН-308 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Ибатуллин И.Т. |  |  |  |
| Консультант | Житников В.П. |  |  |  |
| Принял | Газизов Р.К. |  |  |  |

**Уфа 2014**

ФГБОУ ВПО «Уфимский государственный авиационный технический университет»

Кафедра ВВТиС

**ЗАДАНИЕ**

на курсовую работу по дисциплине

**«Операционные сети и Системы 2»**

Студент: Ибатуллин Ильнур Тимерьярович Группа МКН-308

Консультант: Житников Владимир Павлович

**1. Тема курсовой работы**

Реализация алгоритма поиска расстояний и путей от каждой вершины до остальных.

**2. Основное содержание**

* 1. Требуется написать алгоритм, который путем удаления вершин, не теряя целостность и «истинность», а затем путем добавления находит полную матрицу смежности. Данный алгоритм предназначен для поиска кратчайшего расстояния и пути в графах, полученных из реальных городов.
  2. Выполнить программную реализацию разработанного алгоритма на языке С++ в среде разработки Microsoft Visual Studio 2013(или 2012).
  3. Оформить пояснительную записку к курсовой работе.

**3. Требования к оформлению материалов работы**

3.1. Требования к программе

* Программа должна быть написана на С++, используя STL, приветствуется хорошее оформление кода, наличие комментариев

3.2. Требования к оформлению пояснительной записки

Пояснительная записка к курсовой работе должна быть оформлена в текстовом процессоре Microsoft Word в соответствии с требованиями стандарта СТО УГАТУ 016-2007 и содержать

* титульный лист,
* задание на курсовую работу,
* содержание,
* введение,
* главу, описывающую особенности программной реализации алгоритма,
* главу, описывающую правила работы с программой и примеры ее работы,
* заключение,
* список литературы,
* приложение, содержащее листинг разработанной программы.

**4.** **Список литературы**

1) А.Д., М. (2002). *Введение в теорию фракталов.* Москва, Ижевск.

2) Квант. (1987). *№ 11*, 21.

3) Фролов, А. Ф. (2003). *Язык С#. Самоучитель.*

Дата выдачи задания Дата окончания

\_\_ \_\_\_\_\_\_\_\_\_ 201\_ г. \_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_ г.

Консультант \_\_\_\_\_\_\_\_\_ Житников В.П.

Принял \_\_\_\_\_\_\_\_\_ Газизов Р.К.

Оглавление

[Аннотация 5](#_Toc346160249)

[1. Теоритические основы 6](#_Toc346160250)

[2. Особенности программной реализации алгоритма 10](#_Toc346160251)

[2.1. Описание классов. 10](#_Toc346160252)

[2.2. Алгоритм работы программы. 10](#_Toc346160253)

[3. Правила работы с программой и примеры ее использования 11](#_Toc346160254)

[Результаты и выводы 14](#_Toc346160255)

[литература 15](#_Toc346160256)

## Аннотация

Курсовая работа состоит из следующих разделов:

* в первом разделе содержатся теоретические сведения о графах, различных свойствах графов. Поста
* второй раздел содержит программную реализацию используемых методов – листинг программы, написанной на языке C++, с подробными комментариями.
* третий раздел включает визуализацию результатов полученных значений, составленную средствами разработки Microsoft Visual Studio
* результаты и выводы

## Теоритические основы

**Графы. Свойства графов**

Граф — основной объект изучения математической теории графов, совокупность непустого множества вершин и наборов пар вершин (связей между вершинами).

Объекты представляются как вершины, или узлы графа, а связи — как дуги, или рёбра. Для разных областей применения виды графов могут различаться направленностью, ограничениями на количество связей и дополнительными данными о вершинах или рёбрах.

Многие структуры, представляющие практический интерес в математике и информатике, могут быть представлены графами. Например, строение Википедии можно смоделировать при помощи ориентированного графа, в котором вершины — это статьи, а дуги (ориентированные рёбра) — гиперссылки (тематическая карта).

Граф, или неориентированный граф G — это упорядоченная пара G := (V, E), где V — это непустое множество вершин или узлов, а E — множество пар (в случае неориентированного графа — неупорядоченных) вершин, называемых рёбрами.

V (а значит и, E, иначе оно было бы мультимножеством) обычно считаются конечными множествами. Многие результаты, полученные для конечных графов, неверны (или каким-либо образом отличаются) для бесконечных графов, поскольку не все утверждения, имеющие место для конечных совокупностей, выполняются в случае бесконечных множеств.

Вершины и рёбра графа называются также элементами графа, число вершин в графе |V| — порядком, число рёбер |E| — размером графа.

Вершины u и v называются концевыми вершинами (или просто концами) ребра e = {u, v}. Ребро, в свою очередь, соединяет эти вершины. Две концевые вершины одного и того же ребра называются соседними.

Два ребра называются смежными, если они имеют общую концевую вершину.

Два ребра называются кратными, если множества их концевых вершин совпадают.

Ребро называется петлёй, если его концы совпадают, то есть e = {v, v}.

Степенью deg(V) вершины V называют количество инцидентных ей рёбер (при этом петли считают дважды).

Вершина называется изолированной, если она не является концом ни для одного ребра; висячей (или листом), если она является концом ровно одного ребра.

Ориентированный граф (сокращённо орграф) G — это упорядоченная пара G := (V, A), где V — непустое множество вершин или узлов, и A — множество (упорядоченных) пар различных вершин, называемых дугами или ориентированными рёбрами.

Дуга — это упорядоченная пара вершин (v, w), где вершину v называют началом, а w — концом дуги. Можно сказать, что дуга v → w ведёт от вершины v к вершине w.

Изоморфные графы

Граф G называется изоморфным графу H, если существует биекция f из множества вершин графа G в множество вершин графа H, обладающая следующим свойством: если в графе G есть ребро из вершины A в вершину B, то в графе H должно быть ребро из вершины f(A) в вершину f(B) и наоборот — если в графе H есть ребро из вершины A в вершину B, то в графе G должно быть ребро из вершины f-1(A) в вершину f-1(B). В случае ориентированного графа эта биекция также должна сохранять ориентацию ребра. В случае взвешенного графа биекция также должна сохранять вес ребра.

Прочие связанные определения

Маршрутом в графе называют конечную последовательность вершин, в которой каждая вершина (кроме последней) соединена со следующей в последовательности вершиной ребром. Цепью называется маршрут без повторяющихся рёбер. Простой цепью называется маршрут без повторяющихся вершин (откуда следует, что в простой цепи нет повторяющихся рёбер).

Ориентированным маршрутом (или путём) в орграфе называют конечную последовательность вершин и дуг, в которой каждый элемент инцидентен предыдущему и последующему.

## 2. Особенности программной реализации алгоритма

### 2.1. Описание первого этапа – удаления вершин.

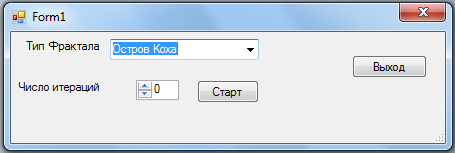
По всему графу ищем вершину с наименьшей степенью вершин. Если таковых несколько, выбираем любой из них. Далее, определяем, с какими вершинами связан. Так как мы собираемся удалить эту вершину, мы должны предусмотреть, чтобы граф при этом не потерял связанность и расстояние между другими вершинами. Допустим, имеется полный связанный граф из 3 вершин. Расстояние из 1 в 2 – 1, из 1 в 3 – 2, из 2 в 3 – 4. Мы хотим удалить вершину 1. При этом мы должны помнить, что кратчайшее расстояние из 2 вершины в 3 равна 3, а не 4. В случае, если мы просто удалим 1 вершину, то потеряем данные об этом. Для этого задействуем правило треугольника. Для любых 2 смежных вершин исходной проверяем неравенство: что короче, расстояние между двумя вершинами, или расстояние между одной вершиной и исходной плюс расстояние между исходной и второй? Именно после этого мы можем быть уверены, что граф после удаления исходной вершины останется связанным, кратчайшие расстояния никуда не исчезнут. Таким образом, мы удалим все вершины, кроме двух последних, нет смысла их удалять. На этом этапе можно сказать, что расстояние между этими двумя вершинами является кратчайшим.

### 2.2. Описание второго этапа – добавление вершин.

После того, как у нас осталось только 2 вершины, наступает пора добавлять вершины обратно в граф. По аналогии математической индукции будем добавлять вершины в обратно порядке удаления. База индукции: 2 вершины, расстояние между ними кратчайшее. Шаг индукции для k: допустим, что для k вершин выполнено условие кратчайших расстояний. Добавим новую вершину. Теперь применим правило треугольника для данной вершины. То есть, проверяем, а не короче ли путь для 2 вершин из базы через новую вершину. Когда всё проверили, мы найдем кратчайшие расстояния от новой вершины до всех остальных. Таким образом, мы построили граф на 1 вершину больше, при этом все накладываемые условия выполняются.

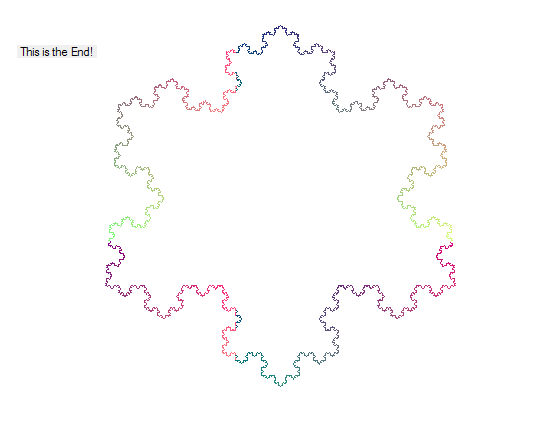
## 3. Правила работы с программой и примеры ее использования

При запуске открывается начальная форма, где можно выбрать, какой из фракталов необходимо нарисовать



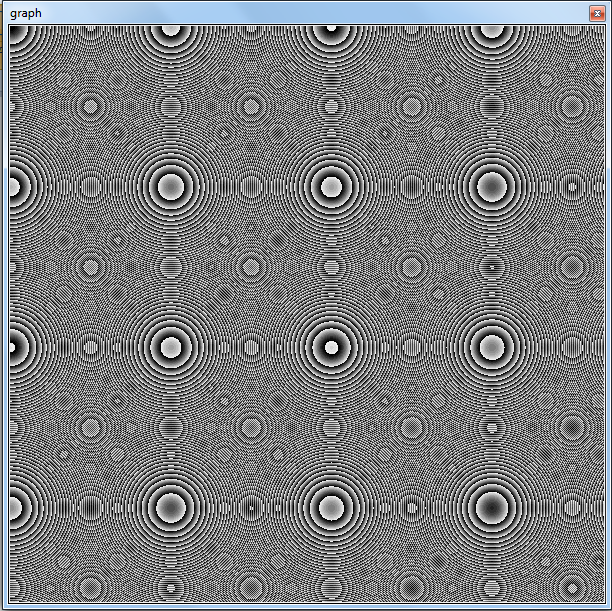
Затем, нажав на кнопку Старт, открывается новое окно, которое не позволяет менять размер окна, где прорисовывается фигура.

Выбрав Остров Коха для 5 итераций, можно увидеть следующее:

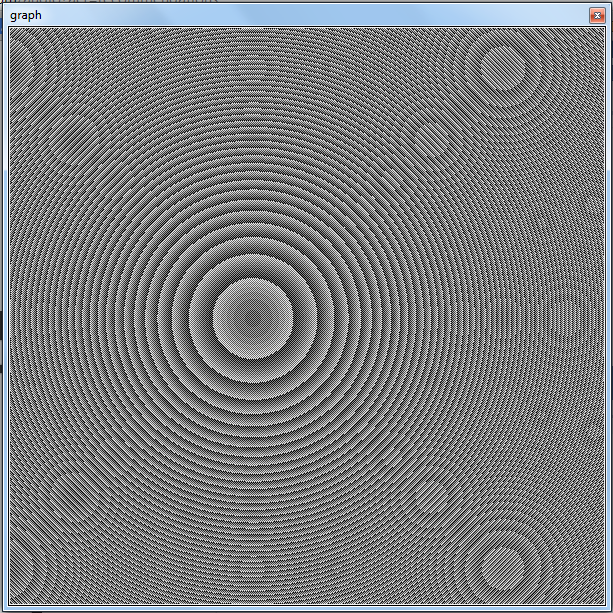


Как видим, отрезки разноцветные, что несколько украшает ломанную.

При выборе пункта «Поверхностный фрактал», можно после запуска увидеть долгую прорисовку точек поверхности, которые рисуются по закону f(x,y)=(x^2+y^2)/S %K, где x и y – это координаты точки, S=20,K=16.



После дорисовки можно выделить прямоугольную область мышкой, которая в дальнейшем прорисуется на полный размер окна.



## Результаты и выводы

При написании данной работы выполнены следующие задания:

1. Разработана программа на С# для визуализации кривой Коха и поверхностного фрактал.
2. Выполнена программная реализация на языке С# в среде разработки Microsoft Visual Studio .NET. Программа содержит следующие обязательные элементы работы с классами:

* конструкторы классов;
* статические компонентные данные и функции;

## литература

А.Д., М. (2002). *Введение в теорию фракталов.* Москва, Ижевск.

Квант. (1987). *№ 11*, 21.

Фролов, А. Ф. (2003). *Язык С#. Самоучитель.*