Дисциплина «Программирование»

Практическое задание №5

24 декабря – 7 декабря

«Фракталы»

Процесс выполнения этого задания состоит из трёх частей:
1) реализация программы, согласно описанным в условии требованиям;
2) оценивание работ других студентов;
3) период «споров».

Период реализации программы:

После выдачи задания Вам необходимо выполнить его и загрузить архив (*.zip) с решением задачи (полностью заархивировать решение, созданное средой разработки) до крайнего срока. В работе строго запрещается указывать ФИО, а также любую другую информацию, которая может выдать авторство работы. В случае выявления факта деанонимизации работы, работа может быть аннулирована.

Период взаимного оценивания:

После окончания срока, отведенного на реализацию программы, начинается период взаимного оценивания. Вам будет необходимо проверить пять работ других студентов, также выполнявших данное задание, согласно критериям оценивания. Проверка осуществляется анонимно: Вы не знаете, чью работу Вы проверяете, также, как и человек, кому принадлежит решение, не знает, кем была проверена его работа. Помимо оценки Вам необходимо указать комментарий к каждому из критериев. В случае, если Вы снижаете балл, необходимо подробно описать, за что именно была снижена оценка. Также рекомендовано писать субъективные комментарии, связанные с тонкостями программной реализации, предлагать автору работу более оптимальные на ваш взгляд решения. Снимать баллы за субъективные особенности реализации запрещено.

Период споров:

По окончании периода взаимного оценивания, в случае если Вы не согласны с оценкой, выставленной одним из проверяющих, Вы можете вступить с этим студентом в анонимный диалог с целью уточнения причин выставления оценки по тому или иному критерию. В случае, если проверяющий не ответил Вам, или вы не пришли к обоюдному решению об изменении оценки, Вы можете поставить флаг, и работа будет рассмотрена одним из преподавателей.

Флаги, поставленные без предварительного обсуждения с проверяющим или после окончания периода выставления флагов, будут отклонены.

Также допустима перепроверка Вашей работы преподавателем. В таком случае оценки других проверявших работу не учитываются.

Дедлайн загрузки работы: 7 декабря 23:59
Дедлайн проверки: 10 декабря 23:59
Дедлайн обсуждения оценок: 12 декабря 23:59
Дедлайн выставления флагов: 13 декабря 23:59

Возможность поздней сдачи работы в этом задании предоставляться не будет.

Оценивание:
$О_{итог} = 0,8 * О_{задание} + 0,2 * О_{проверки}$, где $О_{задание}$ – неокруглённая десятибалльная оценка за решение задания выставленная проверяющими с учётом возможной перепроверки преподавателем, а $О_{проверки}$ неокруглённая десятибалльная оценка, выставленная студентами, чьи работы вы проверяли. Также в случае, если преподаватель обнаружит, что Вы проверили работы некачественно к Вам могут быть применены санкции в виде штрафа до 3 десятичных баллов от $О_{итог}$ (в таком случае оценка вычисляется по формуле $О_{итог} = О_{задание}$ – Штраф).

Необходимо разработать Windows Forms приложение – «Фракталы».

Алгоритм работы приложения:

Разработать оконное приложение Windows Forms App (.Net Core) позволяющее:

1. Отрисовывать пять видов фракталов. Описания фракталов представлены ниже.
2. Предоставлять пользователю выбор текущего фрактала для отрисовки.
3. Предоставлять пользователю возможность устанавливать количество шагов рекурсии (её глубину - количество рекурсивных вызовов). При изменении глубины рекурсии фрактал должен быть автоматически перерисован. Следите за переполнением стека.
4. Сообщать о некорректном вводе данных, противоречивых или недопустимых значениях данных и других нештатных ситуациях во всплывающих окнах типа MessageBox.

5. [Дополнительный функционал] Автоматически перерисовывать фрактал при изменении размеров окна. Окно обязательно должно быть масштабируемым. Вы можете задать минимальный и максимальный размер окна. Максимальным считается размер окна, соответствующий

размеру экрана, а минимальным размером окна считается половинный размер экрана (как по длине, так и по ширине).

6. [Дополнительный функционал] Предоставлять пользователю возможность выбора двух цветов startColor и endColor. Цвет startColor используется для отрисовки элементов первой итерации, цвет endColor - для отрисовки элементов последней итерации. Цвета для промежуточных итераций должны вычисляться с использованием линейного градиента[1].
   Например,

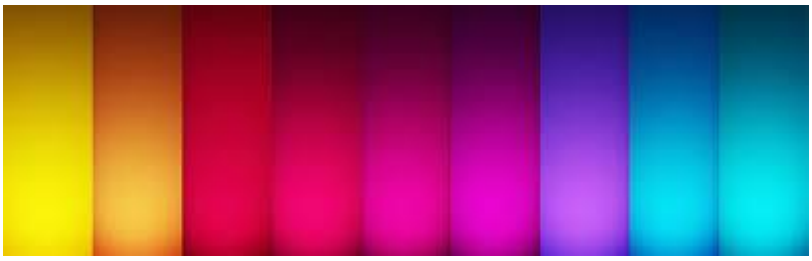

Рис. 1. Градиент цвета

   Желтый цвет используется для отрисовки элементов первой итерации. Синий - для отрисовки элементов последней итерации.

   Промежуточные цвета вычисляются исходя из начального и конечного значений цвета и номера итерации **(не генерируются случайным образом).**

7. [Дополнительный функционал] Предусмотреть возможность изменения масштаба фрактала для его детального просмотра. Увеличение должно быть 2, 3 и 5-кратным
8. [Дополнительный функционал] Предусмотреть возможность перемещения изображения, в т.ч. при увеличенном изображении (пункт 8).
9. [Дополнительный функционал] Должна быть предусмотрена возможность сохранения фрактала в виде картинки (формат выбрать самостоятельно).

Для того, чтобы выделить базовую структуру алгоритма построения фракталов следует разбить процедуру рисования на небольшие действия, и те из них, которые окажутся одинаковыми для разных фракталов, поместить в базовый (родительский) класс. Различающиеся действия поместить в производные классы, создав по одному производному классу для каждого фрактала.

Обязательными полями базового класса «фрактал» следует считать максимальную глубину рекурсии, а также метод для отрисовки фрактала, который должен быть переопределён в каждом из классов наследников. UPD:

[1] http://qaru.site/questions/222000/generate-color-gradient-in-c

Ранее в обязательные поля входила длина отрезка, допускается и не является ошибкой реализация без него.

В зависимости от типа фрактала в классы-наследники следует добавить необходимые члены класса, указанные при описании фрактала. Допустимо добавлять другие члены класса.

Ограничения:

1. В программной реализации не использовать вспомогательные компоненты и сторонние библиотеки, не входящие в стандартную библиотеку.

2. Не использовать массивы типа **object[]**.

Описания фракталов:

Фракталами называют математические множества, обладающие свойством самоподобия: любая часть фрактала подобна всему фракталу целиком. Термин «фрактал» был введён Бенуа Мандельбротом в 1975 году и получил широкую известность с выходом в 1977 году его книги «Фрактальная геометрия природы». Однако многие фракталы были известны и ранее.

1. ***Обдуваемое ветром фрактальное дерево.***

Фрактальное дерево иногда также называют Пифагоровым деревом. Первые итерации построения этого фрактала показаны на рис. 2.

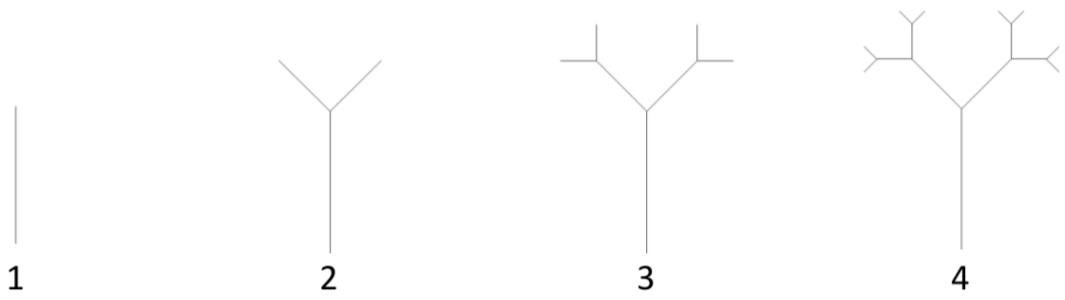

Рис. 2. Первые итерации построения фрактального дерева.

Процесс построения обычного фрактального дерева начинается с рисования вертикальной линии определённой длины. Затем к верхнему концу этой линии добавляют две линии меньшей длины, направленные под углом 45° направо и налево от направления исходной линии. И далее процесс продолжается – на концах каждой из линий добавляют ещё две линии и так далее. Теоретически, такой процесс построения фрактала должен продолжаться бесконечно. На практике его обычно ограничиваются конечным числом этапов (рис. 3).

Если использовать углы другой величины, то можно построить **обдуваемое ветром дерево Пифагора** (рис. 4).

Алгоритм :

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.

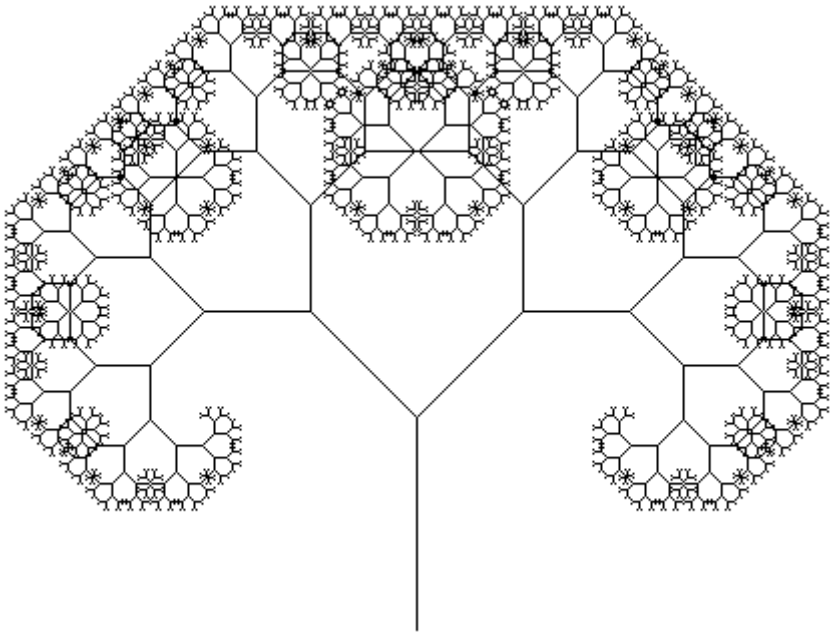

Рис. 3. Фрактальное дерево.

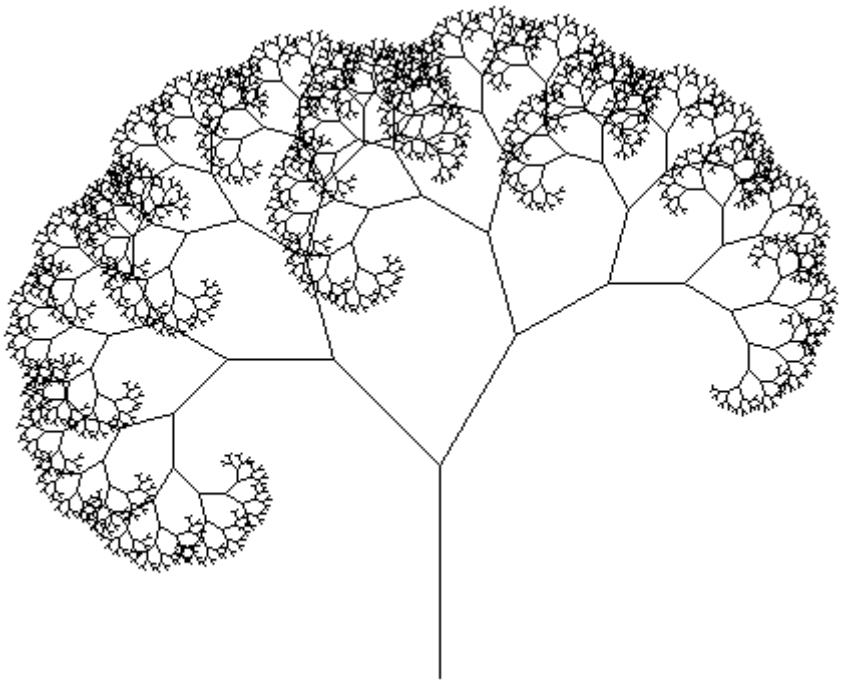

Рис. 4. Обдуваемое ветром фрактальное дерево.

При реализации фрактала учесть, что пользователь должен иметь возможность:

- задавать коэффициент, определяющий отношение длин отрезков на текущей и последующей итерации (с учетом разумных ограничений)
- угол наклона первого отрезка (с учетом разумных ограничений)
- угол наклона второго отрезка (с учетом разумных ограничений)

2. Кривая Коха (рис. 5).

Для построения этой кривой следует взять отрезок и заменить его центральную треть ломаной, повторяющей форму равностороннего треугольника. Затем, каждый из получившихся отрезков подвергается такому же преобразованию.

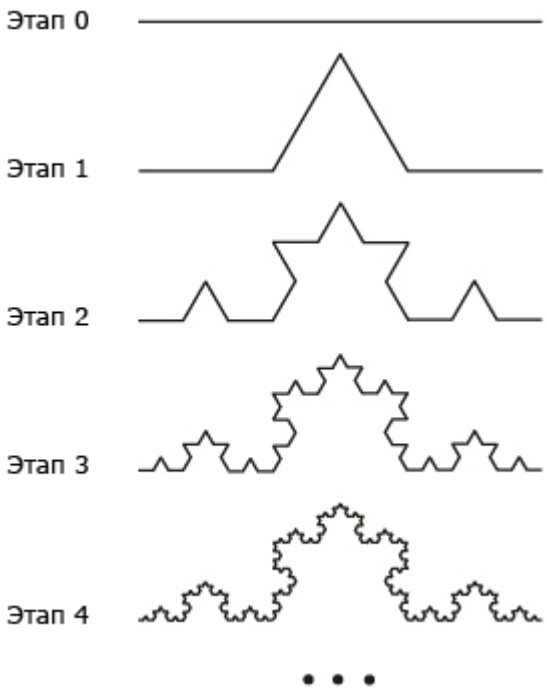

Рис.5. Первые итерации построения кривой Коха.

3. Ковер Серпинского (рис. 6)

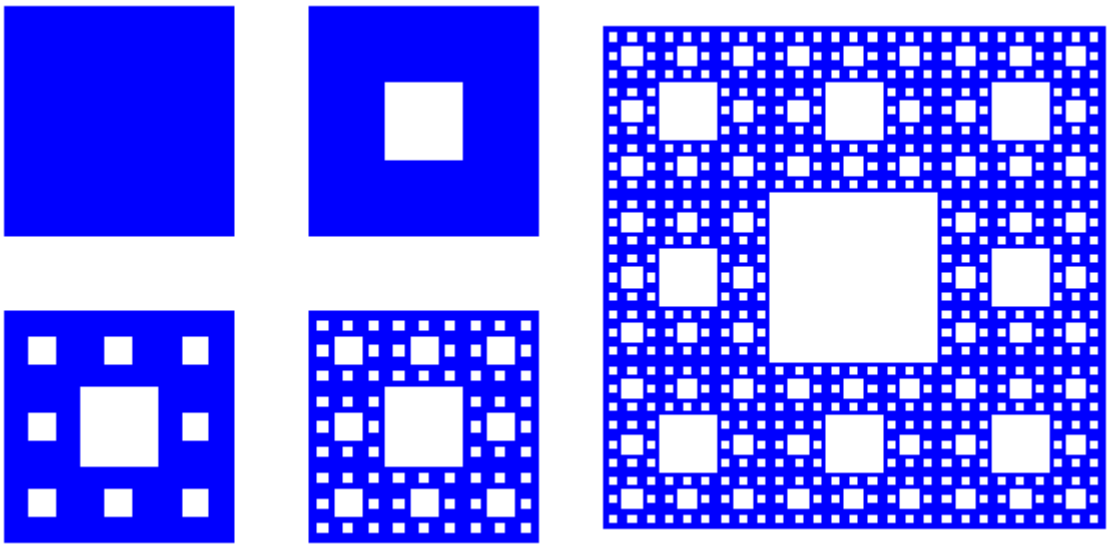

Рис. 6. Первые итерации построения ковра Серпинского

Построение Ковра Серпинского начинается с квадрата. На первом шаге квадрат разбивается сеткой 3 × 3 ячейки и центральная ячейка закрашивается. Затем процесс рекурсивно повторяют для 8 оставшихся ячеек.

4. Треугольник Серпинского

Чтобы получить этот фрактал, нужно взять равносторонний треугольник, затем достроить в нём треугольник, образованный средними линиями, затем в каждом из трех угловых треугольников сделать то же самое, и т. д. (рис. 7).

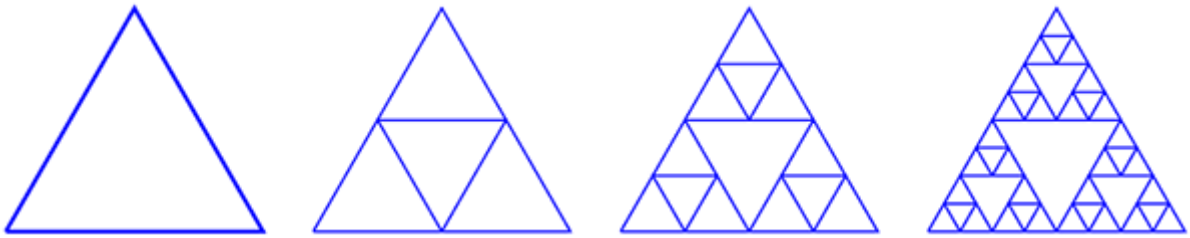

Рис. 7. Первые итерации другого способа построения треугольника Серпинского.

5. Множество Кантора

Способ построения этого множества следующий. Берётся отрезок прямой единичной длины (*[0,1]*). Затем он делится на три равные части, и вынимается средний отрезок (*[1/3, 2/3]*). Это первый шаг итерационной процедуры. На втором шаге подобной процедуре деления на три равные части и последующего удаления середины подвергается каждый из двух оставшихся отрезков. На рис. 8 первые шесть шагов процедуры.

Рис. 8. Множество Кантора

При реализации фрактала учесть, что пользователь должен иметь возможность определять расстояние между отрезками, отрисованными на разных итерациях.

Дополнительные требования (критерии оценивания):

1. Для проверки в систему PeerGrade должен быть загружен архив с решением. Ожидается, что проверка работы будет проводиться, в среде разработки Visual Studio 2019, поэтому в случае выполнения задания с использованием другой среды разработки настоятельно рекомендуется

проверить возможность открытия и запуска проекта в этой среде разработки.

2. Текст программы должен быть отформатирован согласно кодстайлу языка C#[2]. Для автоматического форматирования в среде Visual Studio достаточно нажать Ctrl+K, D. Идентификаторы должны соответствовать соглашению о именовании C#[3]. Основным требованием, которое требуется соблюдать в данном задании – это написание имён локальный переменных с использованием camelCasing, а при именовании методов и типов PascalCasing.
3. Программа не должна завершаться аварийно (или уходить в бесконечный цикл) при любых входных данных. При некорректных входных данных программа должна выводить сообщение об ошибке и запрашивать ввод заново.
4. Программа должна быть декомпозирована. Каждый из логических блоков должен быть выделен в отдельный метод. Не строго, но желательно, чтобы каждый метод по длине не превышал 40 строк.
5. Интерфейс программы должен быть понятен. Пользователю должны выводиться подсказки о возможных дальнейших действиях и иные необходимые сообщения. Предполагается, что для успешного использования программы не требуется обращения к исходному коду программы.
6. Текст программы должен быть документирован. Необходимо писать, как комментарии перед методами, так и комментарии, поясняющие написанный внутри метода код. Названия переменных и методов должны быть на английском языке и отражать суть хранимых значений / выполняемых действий.
7. В работе должны быть представлены все 5 типов фракталов.
8. Существует базовый класс фрактал и пять наследников, т.е. соблюдена структура, представленная в задании.
9. Для получения отличной оценки более 8 студенту необходимо реализовать дополнительный функционал, описанный в п.5 – п.9 основного задания.
10. Также оценивается общее впечатление, которое производит работа (как с точки зрения пользовательского интерфейса, так и с точки зрения написания кода программы). Эта часть оценки остаётся на усмотрение проверяющего.

---

[2] https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions

[3] https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/naming-guidelines