Фрактальное множество Жюлиа - 4

Введение

Отображение $f(z)$, определенное на комплексных числах имеет вид

$$f(z) = cz(1 - z),$$

где c комплексная постоянная. Последовательность z_n определим соотношением

$$z_{n+1} = f(z_n)$$

и начальным условием z_0.

Множеством Жюлиа J отображения $f(z)$ назовем множество таких начальных условий z_0 (комплексных чисел), что z_n не стремится к бесконечности.

На практике, построение множества J в точности невозможно, но можно построить его приближение. Обозначим через $J_N(R)$ множество таких комплексных чисел z_0, что в последовательности z_n первые N членов удовлетворяют соотношению $|z_k| < R$ $(k = 0, 1, \ldots, N)$. При подходящем выборе R и достаточно большом N множества $J_N(R)$ и J будут очень похожи.

Задания

1. Напишите программу, которая при заданных значениях параметров c (например, $1 + i$ или другое комплексное число), $R > 0$ и натуральном N на комплексной плоскости рисует множество $J_N(R)$. Для этого необходимо перебрать (с небольшим шагом) всевозможные комплексные значения z_0, лежащие в круге радиуса R и центром в начале координат. Для каждой точки необходимо проверить, что первые N элементов последовательности z_n тоже лежат внутри этого же круга. Если это условие выполняется, то по определению текущая точка z_0 принадлежит множеству $J_N(R)$ — отмечаем ее на рисунке одним цветом, если же какой-то из элементов z_n лежит вне этого круга (то есть условие нарушилось), то соответствующую точку z_0 отмечаем другим цветом (возможно, в зависимости от номера n элемента z_n, который первый вышел из круга).

2. Определите „эффективное“ значение R. Если R слишком большое, то программа будет работать довольно долго — много перебирать. С другой стороны, если взять R слишком маленьким, то можно потерять часть множества. Определите опытным путем „наилучшее“ значение R. Для этого при достаточно больших значениях N (порядка 100–500) построить множества $J_N(R)$ при разных значениях R. Нам нужно наименьшее из таких значений R, что его увеличение не изменяет получившийся рисунок.

3. Используя программу, подоберите несколько значений параметра c (помните, что он комплексный), чтобы построенное множество обладало фрактальной природой. Имейте ввиду, что при разных c „эффективное“ значение R может отличаться.

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая изображение множества Жюлиа ($J_N(R)$), в зависимости от c, N, R.

2. Сделайте возможность сохранения построенной картинки в jpeg-файл.

3. Дайте возможность пользователю изменять параметры c (не забывайте, что он комплексный), R и N.

4. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.

5. Сделайте картинку разноцветной. Пусть точки остающиеся внутри круга радиуса R будут, например, черными. А покинувшие его на шаге n получают цвет в зависимости от n.

6. Определенное опытным путем „эффективное“ значение R.

7. Определенные опытным путем значения c, при которых наблюдается наиболее интересная (с вашей точки зрения) картинка.

Фрактальное множество Жюлиа - 5

Введение

Отображение $f(z)$, определенное на комплексных числах имеет вид

$$f(z) = z(1 - z) + c,$$

где c комплексная постоянная. Последовательность z_n определим соотношением

$$z_{n+1} = f(z_n)$$

и начальным условием z_0.

Множеством Жюлиа J отображения $f(z)$ назовем множество таких начальных условий z_0 (комплексных чисел), что z_n не стремится к бесконечности.

На практике, построение множества J в точности невозможно, но можно построить его приближение. Обозначим через $J_N(R)$ множество таких комплексных чисел z_0, что в последовательности z_n первые N членов удовлетворяют соотношению $|z_k| < R$ $(k = 0, 1, \ldots, N)$. При подходящем выборе R и достаточно большом N множества $J_N(R)$ и J будут очень похожи.

Задания

1. Напишите программу, которая при заданных значениях параметров c (например, $-0,1+0,1i$ или другое комплексное число), $R > 0$ и натуральном N на комплексной плоскости рисует множество $J_N(R)$. Для этого необходимо перебрать (с небольшим шагом) всевозможные комплексные значения z_0, лежащие в круге радиуса R и центром в начале координат. Для каждой точки необходимо проверить, что первые N элементов последовательности z_n тоже лежат внутри этого же круга. Если это условие выполняется, то по определению текущая точка z_0 принадлежит множеству $J_N(R)$ — отмечаем ее на рисунке одним цветом, если же какой-то из элементов z_n лежит вне этого круга (то есть условие нарушилось), то соответствующую точку z_0 отмечаем другим цветом (возможно, в зависимости от номера n элемента z_n, который первый вышел из круга).

2. Определите „эффективное“ значение R. Если R слишком большое, то программа будет работать довольно долго — много перебирать. С другой стороны, если взять R слишком маленьким, то можно потерять часть множества. Определите опытным путем „наилучшее“ значение R. Для этого при достаточно больших значениях N (порядка 100–500) построить множества $J_N(R)$ при разных значениях R. Нам нужно наименьшее из таких значений R, что его увеличение не изменяет получившийся рисунок.

3. Используя программу, подоберите несколько значений параметра c (помните, что он комплексный), чтобы построенное множество обладало фрактальной природой. Имейте ввиду, что при разных c „эффективное“ значение R может отличаться.

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая изображение множества Жюлиа ($J_N(R)$), в зависимости от c, N, R.

2. Сделайте возможность сохранения построенной картинки в jpeg-файл.

3. Дайте возможность пользователю изменять параметры c (не забывайте, что он комплексный), R и N.

4. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.

5. Сделайте картинку разноцветной. Пусть точки остающиеся внутри круга радиуса R будут, например, черными. А покинувшие его на шаге n получают цвет в зависимости от n.

6. Определенное опытным путем „эффективное“ значение R.

7. Определенные опытным путем значения c, при которых наблюдается наиболее интересная (с вашей точки зрения) картинка.

Фрактальное множество Жюлиа - 6

Введение

Отображение $f(z)$, определенное на комплексных числах имеет вид

$$f(z) = z^5 + c,$$

где c комплексная постоянная. Последовательность z_n определим соотношением

$$z_{n+1} = f(z_n)$$

и начальным условием z_0.

Множеством Жюлиа J отображения $f(z)$ назовем множество таких начальных условий z_0 (комплексных чисел), что z_n не стремится к бесконечности.

На практике, построение множества J в точности невозможно, но можно построить его приближение. Обозначим через $J_N(R)$ множество таких комплексных чисел z_0, что в последовательности z_n первые N членов удовлетворяют соотношению $|z_k| < R$ $(k = 0, 1, \ldots, N)$. При подходящем выборе R и достаточно большом N множества $J_N(R)$ и J будут очень похожи.

Задания

1. Напишите программу, которая при заданных значениях параметров c (например, $-0,1+0,1i$ или другое комплексное число), $R > 0$ и натуральном N на комплексной плоскости рисует множество $J_N(R)$. Для этого необходимо перебрать (с небольшим шагом) всевозможные комплексные значения z_0, лежащие в круге радиуса R и центром в начале координат. Для каждой точки необходимо проверить, что первые N элементов последовательности z_n тоже лежат внутри этого же круга. Если это условие выполняется, то по определению текущая точка z_0 принадлежит множеству $J_N(R)$ — отмечаем ее на рисунке одним цветом, если же какой-то из элементов z_n лежит вне этого круга (то есть условие нарушилось), то соответствующую точку z_0 отмечаем другим цветом (возможно, в зависимости от номера n элемента z_n, который первый вышел из круга).

2. Определите „эффективное“ значение R. Если R слишком большое, то программа будет работать довольно долго — много перебирать. С другой стороны, если взять R слишком маленьким, то можно потерять часть множества. Определите опытным путем „наилучшее“ значение R. Для этого при достаточно больших значениях N (порядка 100–500) построить множества $J_N(R)$ при разных значениях R. Нам нужно наименьшее из таких значений R, что его увеличение не изменяет получившийся рисунок.

3. Используя программу, подоберите несколько значений параметра c (помните, что он комплексный), чтобы построенное множество обладало фрактальной природой. Имейте ввиду, что при разных c „эффективное“ значение R может отличаться.

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая изображение множества Жюлиа ($J_N(R)$), в зависимости от c, N, R.

2. Сделайте возможность сохранения построенной картинки в jpeg-файл.

3. Дайте возможность пользователю изменять параметры c (не забывайте, что он комплексный), R и N.

4. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.

5. Сделайте картинку разноцветной. Пусть точки остающиеся внутри круга радиуса R будут, например, черными. А покинувшие его на шаге n получают цвет в зависимости от n.

6. Определенное опытным путем „эффективное“ значение R.

7. Определенные опытным путем значения c, при которых наблюдается наиболее интересная (с вашей точки зрения) картинка.

Фрактальное множество Мандельброта - 4

Введение

Отображение $f(z)$, определенное на комплексных числах имеет вид $f(z) = z^5 + c$, где c комплексная постоянная. Последовательность z_n определим соотношением

$$z_{n+1} = f(z_n)$$

и начальным условием z_0. Понятно, что последовательность z_n зависит от c.

Множеством Мандельброта M отображения $f(z)$ назовем множество таких значений параметра c (комплексных чисел), что z_n не стремится к бесконечности.

На практике, построение множества M в точности невозможно, но можно построить его приближение. Обозначим через $M_N(R)$ множество таких комплексных чисел c, что в последовательности z_n первые N членов удовлетворяют соотношению $|z_k| < R$ $(k = 0, 1, \ldots, N)$. При подходящем выборе R и достаточно большом N множества $M_N(R)$ и M будут очень похожи.

Задания

1. Напишите программу, которая при заданных значениях параметров z_0 (например, $z_0 = 0$ или другое комплексное число), $R > 0$ и натуральном N на комплексной плоскости рисует множество $M_N(R)$. Для этого необходимо перебрать (с небольшим шагом) всевозможные комплексные значения c, лежащие в некотором квадрате $-a < Re\, c < a$, $-a < Im\, c < a$ (a — это новый параметр). Для каждой точки c необходимо проверить, что первые N элементов последовательности z_n тоже лежат внутри круга радиуса R и с центром в нуле. Если это условие выполняется, то по определению текущая точка c принадлежит множеству $M_N(R)$ — отмечаем ее на рисунке одним цветом, если же какой-то из элементов z_n лежит вне этого круга (то есть условие нарушилось), то соответствующую точку c отмечаем другим цветом.

2. Определите „эффективные“ значения a и R. Если эти параметры слишком большие, то программа будет работать довольно долго — много перебирать. С другой стороны, если взять их слишком маленькими, то можно потерять часть множества. Определите опытным путем „наилучшие“ значения для a и R. Для этого при достаточно больших значениях N (порядка 100–500) построить множества $M_N(R)$ при разных значениях a и R. Нам нужно наименьшее из таких значений, что их увеличение не изменяет получившийся рисунок.

3. Используя программу, подоберите несколько значений параметра z_0 (помните, что он комплексный), чтобы построенное множество обладало фрактальной природой. Имейте ввиду, что при разных z_0 „эффективные“ значения R и a могут отличаться.

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая изображение множества Манельброта ($M_N(R)$), в зависимости от z_0, N, R, a.

2. Определенные опытным путем „эффективные“ значения a и R.

3. Определенные опытным путем значения z_0, при которых наблюдается наиболее интересная (с вашей точки зрения) картинка.

4. Сделайте возможность сохранения построенной картинки в jpeg-файл.

5. Дайте возможность пользователю изменять параметры z_0 (не забывайте, что он комплексный), a, R и N.

6. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.

7. Сделайте картинку разноцветной. Пусть точки, для которых последовательность остается внутри круга радиуса R, будут, например, черными. А остальные получают цвет в зависимости от номера шага n, что впервые $|z_n| > R$.

Фрактальное множество Мандельброта - 5

Введение

Отображение $f(z)$, определенное на комплексных числах имеет вид $f(z) = z(1 - z) + c$, где c комплексная постоянная. Последовательность z_n определим соотношением

$$z_{n+1} = f(z_n)$$

и начальным условием z_0. Понятно, что последовательность z_n зависит от c.

Множеством Мандельброта M отображения $f(z)$ назовем множество таких значений параметра c (комплексных чисел), что z_n не стремится к бесконечности.

На практике, построение множества M в точности невозможно, но можно построить его приближение. Обозначим через $M_N(R)$ множество таких комплексных чисел c, что в последовательности z_n первые N членов удовлетворяют соотношению $|z_k| < R$ $(k = 0, 1, \ldots, N)$. При подходящем выборе R и достаточно большом N множества $M_N(R)$ и M будут очень похожи.

Задания

1. Напишите программу, которая при заданных значениях параметров z_0 (например, $z_0 = 0$ или другое комплексное число), $R > 0$ и натуральном N на комплексной плоскости рисует множество $M_N(R)$. Для этого необходимо перебрать (с небольшим шагом) всевозможные комплексные значения c, лежащие в некотором квадрате $-a < Re\, c < a$, $-a < Im\, c < a$ (a — это новый параметр). Для каждой точки c необходимо проверить, что первые N элементов последовательности z_n тоже лежат внутри круга радиуса R и с центром в нуле. Если это условие выполняется, то по определению текущая точка c принадлежит множеству $M_N(R)$ — отмечаем ее на рисунке одним цветом, если же какой-то из элементов z_n лежит вне этого круга (то есть условие нарушилось), то соответствующую точку c отмечаем другим цветом.

2. Определите „эффективные“ значения a и R. Если эти параметры слишком большие, то программа будет работать довольно долго — много перебирать. С другой стороны, если взять их слишком маленькими, то можно потерять часть множества. Определите опытным путем „наилучшие“ значения для a и R. Для этого при достаточно больших значениях N (порядка 100–500) построить множества $M_N(R)$ при разных значениях a и R. Нам нужно наименьшее из таких значений, что их увеличение не изменяет получившийся рисунок.

3. Используя программу, подоберите несколько значений параметра z_0 (помните, что он комплексный), чтобы построенное множество обладало фрактальной природой. Имейте ввиду, что при разных z_0 „эффективные“ значения R и a могут отличаться.

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая изображение множества Манельброта ($M_N(R)$), в зависимости от z_0, N, R, a.

2. Определенные опытным путем „эффективные“ значения a и R.

3. Определенные опытным путем значения z_0, при которых наблюдается наиболее интересная (с вашей точки зрения) картинка.

4. Сделайте возможность сохранения построенной картинки в jpeg-файл.

5. Дайте возможность пользователю изменять параметры z_0 (не забывайте, что он комплексный), a, R и N.

6. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.

7. Сделайте картинку разноцветной. Пусть точки, для которых последовательность остается внутри круга радиуса R, будут, например, черными. А остальные получают цвет в зависимости от номера шага n, что впервые $|z_n| > R$.

Снежинка Коха

Введение

Снежинка Коха — это фрактальная кривая, которая строится следующим образом. Берем начальный отрезок и делим отрезок на три равных части. На средней части строим правильный треугольник, а саму среднюю часть удаляем. В результате у нас получилось 4 отрезка длины 1/3 каждый. С каждым из них повторяем ту же самую процедуру.

В результате, при бесконечном числе шагов, получается фрактальная кривая которая носит название Снежинка Коха. Разумеется, бесконечное число шагов сделать не удастся. Однако, если число шагов n будет достаточно большим, то изображение на экране будет очень похоже на настоящую Снежинку Коха.

Алгоритм построения этого фрактала может быть, например, рекурсивным. Понадобится функция, скажем $Koh(A, B, n)$, которая строит n-шаговый фрактал на отрезке от A до B (это не числа, а точки на поскости). Если $n = 0$, то просто рисуем отрезок AB, в ином случае последовательно запускаем эту же функцию для всех четырех отрезков и значения $n - 1$.

Задания

1. Напишите программу, которая на координатной плоскости, стартуя с отрезка (0,0)–(1,0), рисует Снежинку Коха в зависимости от заданного числа шагов n. Пусть весь рисунок располагается в квадрате, например, 400×400 точек на экране. Левый нижний угол пусть имеет координаты $(0, 0)$, а правый нижний — $(x, 0)$, где, для начала, $x = 1$.

2. Сделайте возможность изменения масштаба видимой области. То есть, возможность задать x-координату правому-нижнему углу (y-координата там всегда ноль). Таким образом, мы сумеем рассмотреть поближе часть фрактала. Реализуйте хотя бы значения $x = 1$, $x = 1/2$, $x = 1/3$, $x = 1/4$.

3. Если увеличивать число n, то программа будет работать дольше и дольше. Однако, начиная с некоторого шага, новые отрезочки будут иметь очень маленький размер и будут не видны. Подберите опытным путем такое значение n, что его увеличение не изменяет видимое изображение. Сделайте это для масштаба $1 : 1$ и для двух-, трех- и четырехкратного увеличения.

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая Снежинку Коха (или ее часть, если задан определенный масштаб), в зависимости от числа шагов.

2. „Эффектиные“ значения числа шагов для разных масштабов.

3. Сделайте возможность сохранения построенной картинки в jpeg-файл.

4. Дайте возможность пользователю изменять параметры: n, x.

5. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.

6. Определите „эффективное“ значение n в зависимости от масштаба. Найдите формулу (не страшно если она будет не самой оптимальной) и реализуйте ее в программе.

7. Сделайте настоящую снежинку: постройте сразу несколько таких фракталов, начинающихся на сторонах треугольника или другой фигуры.

Треугольник Серпинского

Введение

Треугольник Серпинского — это фрактальное множество, которое строится следующим образом. Берем начальный треугольник. Делим каждую его сторону пополам и соединяем середины сторон. Получается четыре треугольника. Выкидываем средний, а с оставшимися тремя проделываем такую же процедуру.

В результате, при бесконечном числе шагов, получается фрактальное множество, которое носит название Треугольник Серпинского. Разумеется, бесконечное число шагов сделать не удастся. Однако, если число шагов n будет достаточно большим, то изображение на экране будет очень похоже на настоящие Треугольник Серпинского.

Алгоритм построения этого фрактала может быть, например, рекурсивным. Понадобится функция, скажем $TS(A, B, C, n)$, которая строит n-шаговый фрактал на треугольнике ABC. Если $n = 0$, то просто рисуем треугольник ABC, в ином случае последовательно запускаем эту же функцию для всех трех оставшихся треугольников и $n - 1$.

Задания

1. Напишите программу, которая на координатной плоскости, стартуя с треугольника ABC (например, $A(0, 0)$, $B(1, 0)$, $C(1/2, 1)$, рисует Треугольник Серпинского в зависимости от заданного числа шагов n. Пусть весь рисунок располагается в квадрате, например, 400×400 точек на экране. Левый нижний угол пусть имеет координаты $(0, 0)$, а правый верхний — (z, z), где, для начала, $z = 1$.

2. Сделайте возможность изменения масштаба видимой области. То есть, возможность задать параметр z, определяющий координаты правого-верхнего угла. Таким образом, мы сумеем рассмотреть поближе часть фрактала. Реализуйте хотя бы значения $z = 1$, $z = 1/2$, $z = 1/3$, $z = 1/4$.

3. Если увеличивать число n, то мноджество будет “исчезать” с экрана. Подберите опытным путем такое значение n, что множество уже выглядит достаточно сложным, но при этом еще хорошо заметным. Сделайте это для масштаба $1 : 1$ и для двух-, трех- и четырехкратного увеличения.

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая Треугольник Серпинского (или ее часть, если задан определенный масштаб), в зависимости от числа шагов.

2. „Эффектиные“ значения числа шагов для разных масштабов.

3. Сделайте возможность сохранения построенной картинки в jpeg-файл.

4. Дайте возможность пользователю изменять параметры: n, z.

5. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.

6. Определите „эффективное“ значение n в зависимости от масштаба. Найдите формулу (не страшно если она будет не самой оптимальной) и реализуйте ее в программе.

7. Дайте возможность пользователю выбрать начальный треугольник (точки A, B и C).

Ковер Серпинского

Введение

Ковер Серпинского — это фрактальное множество, которое строится следующим образом. Берем начальный квадрат. Делим каждую его сторону на три равных части, отрезки, соединяющие трети его сторон, делят квадрат на 9 маленьких квадратиков. Выкидываем средний, а с оставшимися восемью проделываем такую же процедуру.

В результате, при бесконечном числе шагов, получается фрактальное множество, которое носит название Ковер Серпинского. Разумеется, бесконечное число шагов сделать не удастся. Однако, если число шагов n будет достаточно большим, то изображение на экране будет очень похоже на настоящий Ковер Серпинского.

Алгоритм построения этого фрактала может быть, например, рекурсивным. Понадобится функция, скажем $CS(A, B, C, D, n)$, которая строит n-шаговый фрактал на квадрате ABC. Если $n = 0$, то просто рисуем квадрат $ABCD$, в ином случае последовательно запускаем эту же функцию для всех восьми оставшихся квадратов и $n - 1$.

Задания

1. Напишите программу, которая на координатной плоскости, стартуя с единичного квадрата, рисует Ковер Серпинского в зависимости от заданного числа шагов n. Пусть весь рисунок располагается в квадрате, например, 400×400 точек на экране. Левый нижний угол пусть имеет координаты $(0, 0)$, а правый верхний — (z, z), где, для начала, $z = 1$.

2. Сделайте возможность изменения масштаба видимой области. То есть, возможность задать параметр z, определяющий координаты правого-верхнего угла. Таким образом, мы сумеем рассмотреть поближе часть фрактала. Реализуйте хотя бы значения $z = 1$, $z = 1/2$, $z = 1/3$, $z = 1/4$.

3. Если увеличивать число n, то программа будет работать дольше и дольше. Однако, начиная с некоторого шага, новые отрезочки будут иметь очень маленький размер и будут не видны. Подберите опытным путем такое значение n, что его увеличение не изменяет видимое изображение. Сделайте это для масштаба $1 : 1$ и для двух-, трех- и четырехкратного увеличения.

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая Ковер Серпинского (или ее часть, если задан определенный масштаб), в зависимости от числа шагов.

2. „Эффектиные“ значения числа шагов для разных масштабов.

3. Сделайте возможность сохранения построенной картинки в jpeg-файл.

4. Дайте возможность пользователю изменять параметры: n, z.

5. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.

6. Определите „эффективное“ значение n в зависимости от масштаба. Найдите формулу (не страшно если она будет не самой оптимальной) и реализуйте ее в программе.

7. Дайте возможность пользователю выбрать начальный четырехугольник (а не только квадрат), т.е. задать точки A, B, C и D.

Кладбище Серпинского

Введение

Кладбище Серпинского — это фрактальное множество, которое строится следующим образом. Берем начальный квадрат. Делим каждую его сторону на три равных части, отрезки, соединяющие трети его сторон, делят квадрат на 9 маленьких квадратиков. Выкидываем четыре угловых, а с оставшимися пятью (они образуют крест) проделываем такую же процедуру.

В результате, при бесконечном числе шагов, получается фрактальное множество, которое носит название Кладбище Серпинского. Разумеется, бесконечное число шагов сделать не удастся. Однако, если число шагов n будет достаточно большим, то изображение на экране будет очень похоже на настоящее Кладбище Серпинского.

Алгоритм построения этого фрактала может быть, например, рекурсивным. Понадобится функция, скажем $CS(A, B, C, D, n)$, которая строит n-шаговый фрактал на квадрате $ABCD$. Если $n = 0$, то просто рисуем квадрат $ABCD$, в ином случае последовательно запускаем эту же функцию для всех пяти оставшихся квадратов и $n - 1$.

Задания

1. Напишите программу, которая на координатной плоскости, стартуя с единичного квадрата, рисует Кладбище Серпинского в зависимости от заданного числа шагов n. Пусть весь рисунок располагается в квадрате, например, 400×400 точек на экране. Левый нижний угол пусть имеет координаты $(0, 0)$, а правый верхний — (z, z), где, для начала, $z = 1$.

2. Сделайте возможность изменения масштаба видимой области. То есть, возможность задать параметры, определяющие координаты левого-нижнего и правого-верхнего угла. Таким образом, мы сумеем рассмотреть поближе часть фрактала.

3. Если увеличивать число n, то программа будет работать дольше и дольше. Однако, начиная с некоторого шага, новые кусочки будут иметь очень маленький размер и будут не видны. Подберите опытным путем такое значение n, что его увеличение не изменяет видимое изображение. Сделайте это для масштаба $1 : 1$ и для двух-, трех- и четырехкратного увеличения.

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая Кладбище Серпинского (или ее часть, если задан определенный масштаб), в зависимости от числа шагов.

2. „Эффектиные“ значения числа шагов для разных масштабов.

3. Сделайте возможность сохранения построенной картинки в jpeg-файл.

4. Дайте возможность пользователю изменять параметры: n.

5. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.

6. Определите „эффективное“ значение n в зависимости от масштаба. Найдите формулу (не страшно если она будет не самой оптимальной) и реализуйте ее в программе.

7. Дайте возможность пользователю выбрать начальный четырехугольник (а не только квадрат), т.е. задать точки A, B, C и D.

Кладбище Серпинского - 2

Введение

Кладбище Серпинского 2 — это фрактальное множество, которое строится следующим образом. Берем начальный квадрат. При каждой вершине строим меньший квадрат со стороной в k раз меньше, а все остальное (в виде креста) выбрасываем. После чего с оставшимимся четырьмя квадратиками проделываем аналогичную процедуру.

В результате, при бесконечном числе шагов, получается фрактальное множество, которое назовем Кладбище Серпинского 2. Разумеется, бесконечное число шагов сделать не удастся. Однако, если число шагов n будет достаточно большим, то изображение на экране будет очень похоже на итоговый фрактал.

Алгоритм построения этого фрактала может быть, например, рекурсивным. Понадобится функция, скажем $CS(A, B, C, D, n)$, которая строит n-шаговый фрактал на квадрате $ABCD$. Если $n = 0$, то просто рисуем квадрат $ABCD$, в ином случае последовательно запускаем эту же функцию для четырех меньших квадратов и $n - 1$.

Задания

1. Напишите программу, которая на координатной плоскости, стартуя с единичного квадрата, рисует Кладбище Серпинского 2 в зависимости от заданного числа шагов n. Пусть весь рисунок располагается в квадрате, например, 400×400 точек на экране. Левый нижний угол пусть имеет координаты $(0, 0)$, а правый верхний — (z, z), где, для начала, $z = 1$.

2. Сделайте возможность изменения масштаба видимой области. То есть, возможность задать параметры, определяющие координаты левого-нижнего и правого-верхнего угла. Таким образом, мы сумеем рассмотреть поближе часть фрактала.

3. Если увеличивать число n, то программа будет работать дольше и дольше. Однако, начиная с некоторого шага, новые кусочки будут иметь очень маленький размер и будут не видны. Подберите опытным путем такое значение n, что его увеличение не изменяет видимое изображение. Сделайте это для масштаба $1 : 1$ и для двух-, трех- и четырехкратного увеличения.

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая Кладбище Серпинского 2 (или ее часть, если задан определенный масштаб), в зависимости от числа шагов.

2. „Эффектиные“ значения числа шагов для разных масштабов.

3. Сделайте возможность сохранения построенной картинки в jpeg-файл.

4. Дайте возможность пользователю изменять параметры: n.

5. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.

6. Определите „эффективное“ значение n в зависимости от масштаба. Найдите формулу (не страшно если она будет не самой оптимальной) и реализуйте ее в программе.

7. Дайте возможность пользователю выбрать начальный четырехугольник (а не только квадрат), т.е. задать точки A, B, C и D.

Дерево Пифагора

Введение

Рассмотрим фрактальное множество, которое строится следующим образом. Берем начальный квадрат. На верхней его стороне, наружу строим прямоугольный треугольник у которого условный "левый" острый угол равен α, а сторона квадрата является гипотенузой. На каждом катете этого треугольника строим квадрат. На противоположной стороне каждого квадрата опять строим прямоугольный треугольник с "левым" углом α, затем опять добавляем два квадрата и так далее.

В результате, при бесконечном числе шагов, получается фрактальное множество, которое носит название Дерево Пифагора. Разумеется, бесконечное число шагов сделать не удастся. Однако, если число шагов n будет достаточно большим, то изображение на экране будет очень похоже на настоящее Дерево Пифагора.

Алгоритм построения этого фрактала может быть, например, рекурсивным. Понадобится функция, скажем $Sqr(A, B, n)$, которая строит n-шаговый фрактал на квадрате, одна из сторон которого — это AB. Если $n = 0$, то просто рисуем этот квадрат, в ином случае пририсовываем треугольник, а затем последовательно запускаем эту же функцию для обоих катетов и $n - 1$.

Задания

1. Напишите программу, которая на координатной плоскости, стартуя с единичного квадрата, рисует Дерево Пифагора в зависимости от заданного числа шагов n и параметра α (например возьмите $\alpha = 45°$). Пусть весь рисунок располагается в квадрате, например, 400×400 точек на экране. Левый нижний угол пусть имеет координаты $(-1, 0)$, правый нижний — $(2, 0)$.

2. Сделайте возможность изменения параметра α. Таким образом, мы сумеем построить не только "классическое" дерево, но и "обдуваемое ветром".

3. Если увеличивать число n, то программа будет работать дольше и дольше. Однако, начиная с некоторого шага, новые фигурки будут иметь очень маленький размер и будут не видны. Подберите опытным путем такое значение n, что его увеличение не изменяет видимое изображение. Сделайте это для разных значений параметра α.

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая Дерево Пифагора (или ее часть, если задан определенный масштаб), в зависимости от числа шагов и параметра α.

2. „Эффектиные“ значения числа шагов для разных параметров.

3. Сделайте возможность сохранения построенной картинки в jpeg-файл.

4. Дайте возможность пользователю изменять параметры: n, α.

5. Сделайте дерево разноцветным. Пусть более-менее' "большие" объекты рисуются одним цветом (например коричневым), а те, что поменьше – зеленым.

6. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.

7. Определите „эффективное“ значение n в зависимости от масштаба. Найдите формулу (не страшно если она будет не самой оптимальной) и реализуйте ее в программе.

H-Дерево

Введение

Рассмотрим фрактальное множество, которое строится следующим образом. Берем начальный отрезок, расположенный горизонтально. Через обе его вершины строим перпендикулярные ему отрезки, так чтобы получилась буква Н. Длины новых отрезков пусть будут меньше в k раз ($k > 1$), а соотношение в котором новые отрезки делятся изначальным равно $q > 0$. Затем с новыми отрезками проделаем такую же процедуру. И так далее.

В результате, при бесконечном числе шагов, получается фрактальное множество, которое назовем Н-Дерево. Разумеется, бесконечное число шагов сделать не удастся. Однако, если число шагов n будет достаточно большим, то изображение на экране будет очень похоже на настоящий фрактал.

Алгоритм построения этого фрактала может быть, например, рекурсивным. Понадобится функция, скажем $H(A, B, n)$, которая строит n-шаговый фрактал на отрезке AB. Если $n = 0$, то просто рисуем этот отрезок, в ином случае последовательно запускаем функцию $H(C_1, C_2, n-1)$ и $H(D_1, D_2, n-1)$, где $C_{1,2}$ и $D_{1,2}$ — это вычисленные вершины следующих отрезков.

Задания

1. Напишите программу, которая на координатной плоскости, стартуя с единичного вертикального отрезка, рисует Н-Дерево в зависимости от заданного числа шагов n и параметров k и q (например возьмите $k = 2$, $q = 1$). Пусть весь рисунок располагается в квадрате, например, 400×400 точек на экране. Левый нижний угол пусть имеет координаты $(-1, -1)$, правый верхний — $(1, 1)$.

2. Сделайте возможность изменения параметров k и q.

3. Если увеличивать число n, то программа будет работать дольше и дольше. Однако, начиная с некоторого шага, новые отрезки будут иметь очень маленький размер и будут не видны. Подберите опытным путем такое значение n, что его увеличение не изменяет видимое изображение. Сделайте это для разных значений параметров.

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая Н-Дерево (или его часть, если задан определенный масштаб), в зависимости от числа шагов и параметров k q.

2. „Эффектиные“ значения числа шагов для разных параметров.

3. Сделайте возможность сохранения построенной картинки в jpeg-файл.

4. Дайте возможность пользователю изменять параметры: n, k, q.

5. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.

6. Определите „эффективное“ значение n в зависимости от масштаба. Найдите формулу (не страшно если она будет не самой оптимальной) и реализуйте ее в программе.

7. Сделайте дерево разноцветным. Пусть более-менее’ “большие” объекты рисуются одним цветом (например коричневым), а те, что поменьше – зеленым. Попробуйте сделать “плавную” зависимость цвета от длины отрезка.

Дерево из палочек №2

Введение

Рассмотрим фрактальное множество, которое строится следующим образом. Берем начальный отрезок, расположенный вертикально. Из точки на "высоте" $1/3$ его длины выпустим отрезок влево, под углом α, из точки на высоте $2/3$ выпустим отрезок вправо под углом β, наконец из вершины выпустим вверх отрезок. Длины новых отрезков возьмем в k раз ($k > 1$) меньше, чем у исходного. Затем с каждым новым отрезком поступаем аналогично. И так далее.

В результате, при бесконечном числе шагов, получается фрактальное множество, которое назовем Дерево из палочек №2. Разумеется, бесконечное число шагов сделать не удастся. Однако, если число шагов n будет достаточно большим, то изображение на экране будет очень похоже на настоящий фрактал.

Алгоритм построения этого фрактала может быть, например, рекурсивным. Понадобится функция, скажем $L(A, B, n)$, которая строит n-шаговый фрактал на отрезке AB. Если $n = 0$, то просто рисуем этот отрезок, в ином случае последовательно запускаем функцию $L(C_1, D_1, n - 1$, $L(C_2, D_2, n - 1)$ и $L(B, D_3, n - 1)$ где $C_{1,2}$ — это треть и две трети отрезка, $D_{1,2,3}$ — вычисленные вершины следующих отрезков.

Задания

1. Напишите программу, которая на координатной плоскости, стартуя с единичного вертикального отрезка, рисует Дерево из палочек №2 в зависимости от заданного числа шагов n и параметров k, α и β (например возьмите $k = 2$, $\alpha = \beta = 45°$). Пусть весь рисунок располагается в квадрате, например, 400×400 точек на экране. Левый нижний угол пусть имеет координаты $(-1, 0)$, правый нижний — $(1, 0)$.

2. Сделайте возможность изменения параметров k, α и β. Таким образом, мы сумеем построить не только "классическое" дерево, но и "обдуваемое ветром".

3. Если увеличивать число n, то программа будет работать дольше и дольше. Однако, начиная с некоторого шага, новые отрезки будут иметь очень маленький размер и будут не видны. Подберите опытным путем такое значение n, что его увеличение не изменяет видимое изображение. Сделайте это для разных значений параметров k, α и β.

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая Дерево из палочек №2 (или его часть, если задан определенный масштаб), в зависимости от числа шагов и параметров k, α и β.

2. „Эффектиные“ значения числа шагов для разных параметров.

3. Сделайте возможность сохранения построенной картинки в jpeg-файл.

4. Дайте возможность пользователю изменять параметры: n, k, α, β.

5. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.

6. Определите „эффективное“ значение n в зависимости от масштаба. Найдите формулу (не страшно если она будет не самой оптимальной) и реализуйте ее в программе.

7. Сделайте дерево разноцветным. Пусть более-менее' "большие" объекты рисуются одним цветом (например коричневым), а те, что поменьше – зеленым.

Треугольник Серпинского №2

Введение

Треугольник Серпинского №2 — это фрактальное множество, которое строится следующим образом. Берем начальный треугольник. Делим каждую его сторону на три части и проводим через трети прямые, параллельные сторонам треугольника. Они разрезают исходный треугольник на 9 других. Из них оставим только только те, которые имеют общие стороны с исходным (их 6), а три других выкидываем. С оставшимися шестью проделываем такую же процедуру.

В результате, при бесконечном числе шагов, получается фрактальное множество, которое назовем Треугольник Серпинского №2. Разумеется, бесконечное число шагов сделать не удастся. Однако, если число шагов n будет достаточно большим, то изображение на экране будет очень похоже на настоящий Треугольник Серпинского №2.

Алгоритм построения этого фрактала может быть, например, рекурсивным. Понадобится функция, скажем $TS(A, B, C, n)$, которая строит n-шаговый фрактал на треугольнике ABC. Если $n = 0$, то просто рисуем треугольник ABC, в ином случае последовательно запускаем эту же функцию для всех шести оставшихся треугольников и $n - 1$.

Задания

1. Напишите программу, которая на координатной плоскости, стартуя с треугольника ABC (например, $A(0, 0)$, $B(1, 0)$, $C(1/2, 1)$, рисует Треугольник Серпинского №2 в зависимости от заданного числа шагов n. Пусть весь рисунок располагается в квадрате, например, 400×400 точек на экране. Левый нижний угол пусть имеет координаты $(0, 0)$, а правый верхний — (z, z), где, для начала, $z = 1$.

2. Сделайте возможность изменения масштаба видимой области. То есть, возможность задать параметр z, определяющий координаты правого-верхнего угла. Таким образом, мы сумеем рассмотреть поближе часть фрактала. Реализуйте хотя бы значения $z = 1$, $z = 1/2$, $z = 1/3$, $z = 1/4$.

3. Если увеличивать число n, то мноджество будет "исчезать" с экрана. Подберите опытным путем такое значение n, что множество уже выглядит достаточно сложным, но при этом еще хорошо заметным. Сделайте это для масштаба $1 : 1$ и для двух-, трех- и четырехкратного увеличения.

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая Треугольник Серпинского №2 (или ее часть, если задан определенный масштаб), в зависимости от числа шагов.

2. „Эффектиные“ значения числа шагов для разных масштабов.

3. Сделайте возможность сохранения построенной картинки в jpeg-файл.

4. Дайте возможность пользователю изменять параметры: n, z.

5. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.

6. Определите „эффективное“ значение n в зависимости от масштаба. Найдите формулу (не страшно если она будет не самой оптимальной) и реализуйте ее в программе.

7. Дайте возможность пользователю выбрать начальный треугольник (точки A, B и C).

Игра в Хаос - 1

Введение

Возьмем на плоскости 3 точки A, B, C. Выберем еще одну точку X_0. Далее будем строить последовательность точек X_1, X_2, ... по следующему правилу. Случайным образом выбираем одну из трех точек A или B или C. И строим точку X_1, как середину между X_0 и выбранной. Затем опять случайно выбираем A или B или C, и стороим X_2 как середину между X_1 и выбранной. И так далее.

Такой процесс мы будем называть игрой в хаос.

Задания

1. Напишите программу, которая реализует игру в хаос, по описаным правилам. Пусть точки A, B, C и X_0 располагаются в квадрате (например, $A(0,0)$, $B(1,0)$, $C(1/2,1)$, $X_0(1/2,1/2)$), например, 400×400 точек на экране. Левый нижний угол пусть имеет координаты $(0,0)$, а правый верхний — $(1,1)$. Программа должна выводить на экран первые n точек (скажем $n = 100000$) последовательности X_1, X_2, ...

2. Как изменится рисунок при изменении X_0? При изменении A, B, C?

3. Реализуйте это же множество, но при других вероятностях выбора точек A или B или C на каждом шаге. Как изменится множество?

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), реализующая игру в хаос, в зависимости от числа шагов n, точек A, B, C, X_0.

2. Ответы на вопросы заданий 2 и 3.

3. Сделайте возможность сохранения построенной картинки в jpeg-файл.

4. Дайте возможность пользователю изменять параметры.

5. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.

6. Классифицируйте получающееся множество. Как бы вы обосновали, что оно именно такое?

7. Дайте пользователю возможность изменять точки A, B и C.

Игра в Хаос - 2

Введение

Возьмем на плоскости 4 точки A, B, C, D. Выберем еще одну точку X_0. Далее будем строить последовательность точек X_1, X_2, ... по следующему правилу. Случайным образом выбираем одну из четырех точек A, B, C или D. И строим точку X_1, как точку на отрезке между X_0 и выбранной, делящая его в соотношении q (считая от X_0). Затем опять случайно выбираем A, B, C или D, и стороим X_2 точку, делящую отрезок между X_1 и выбранной в соотношении q. И так далее.

Такой процесс мы будем называть игрой в хаос.

Задания

1. Напишите программу, которая реализует игру в хаос, по описаным правилам. Пусть точки A, B, C, D являются вершинами единичного квадрата, а X_0 располагается внутри него(например, $X_0(1/2), 1/2))$. Размер всего изображения будет, например, 400×400 точек на экране. Левый нижний угол пусть имеет координаты $(0, 0)$, а правый верхний — (z, z), где, для начала, $z = 1$. Программа должна выводить на экран первые n точек (скажем $n = 100000$) последовательности X_1, X_2, ...

2. Как изменится рисунок при изменении X_0? При изменении q?

3. Реализуйте возможность изменения параметра z.

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), реализующая игру в хаос, в зависимости от числа шагов n, точек A, B, C, X_0.

2. Ответы на вопросы задания 2.

3. Сделайте возможность сохранения построенной картинки в jpeg-файл.

4. Дайте возможность пользователю изменять параметры.

5. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.

6. Классифицируйте получающееся множество. Как бы вы обосновали, что оно именно такое?

7. Дайте пользователю возможность изменять точки A, B, C и D.

Игра в Хаос - 4

Введение

Выберем на плоскости точку X с координатами (x, y). Рассмотрим следующее отображение этой точки. Случайным образом выбираем натуральное число p от 1 до 100. Далее точку Y определяем по правилу

$$Y = A(p)X + b(p),$$

где A — матрица 2×2, а b — вектор, зависящие от p. Затем опять выберем натуральное p и уже точку Y преобразуем по такому правилу. И так далее.

Такой процесс мы будем называть игрой в хаос.

Задания

1. Напишите программу, которая реализует игру в хаос, по описаным правилам. Матрица $A(p)$ имеет вид:

$$A(p) = \begin{pmatrix} 0 & 0.24 \\ 0 & 0.305 \end{pmatrix} \text{ при } p = 1; \qquad A(p) = \begin{pmatrix} 0.72 & 0.034 \\ -0.025 & 0.742 \end{pmatrix} \text{ при } 1 < p \leq 72;$$

$$A(p) = \begin{pmatrix} 0.158 & -0.128 \\ 0.355 & 0.367 \end{pmatrix} \text{ при } 72 < p \leq 86; \qquad A(p) = \begin{pmatrix} 0.339 & 0.369 \\ 0.223 & -0.076 \end{pmatrix} \text{ при } 86 < p \leq 100.$$

Вектор $b(p)$ при этом определяется так:

$$b(p) = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \text{ при } p = 1; \qquad b(p) = \begin{pmatrix} 0.206 \\ 0.254 \end{pmatrix} \text{ при } 1 < p \leq 72;$$

$$b(p) = \begin{pmatrix} 0.138 \\ 0.175 \end{pmatrix} \text{ при } 72 < p \leq 86; \qquad b(p) = \begin{pmatrix} 0.68 \\ 0.083 \end{pmatrix} \text{ при } 86 < p \leq 100.$$

Программа должна отображать первые n (например $n = 10000$) точек. Что получится в итоге?

2. Поизменяйте числовые параметры в матрице A и векторе b. Как будет меняться изображение?

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), реализующая игру в хаос.
2. Сделайте возможность сохранения построенной картинки в jpeg-файл.
3. Дайте возможность пользователю изменять параметры.
4. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного множества.
5. Как будет изменяться изображение при небольшом изменении тех или иных параметров матриц $A(p)$ и векторов $b(p)$?

Бассейны Ньютона - 1

Введение

Рассмотрим уравнение в комплексных числах

$$f(z) = z^3 + c = 0,$$

где c – комплексная постоянная. Метод Ньютона для нахождения решения такого уравнения состоит в последовательном вычислении z_n по правилу

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}$$

при некоторым начальном условии z_0. Предел последовательности – корень уравнения.

Выбор начального z_0 очень важен. Так как функция может иметь несколько нулей, то при различных z_0 метод может сходиться к различным корням. Однако, что за области обеспечат сходимость к тому или иному корню?

Бассейнами Ньютона назовем разбиение комплексной плоскости на множества, соответствующих одному и тому же корню $f(z) = 0$.

На практике, построение такого разбиения в точности невозможно, но можно построить его приближение. Обозначим через $P_i(R)$ множество таких комплексных чисел z_0, что в последовательности z_n найдется элемент, находящийся на расстоянии меньше R от i-го корня уравнения. Эти множества будут хорошим приближением настоящих бассейнов.

Задания

1. Напишите программу, которая при заданных значениях параметров c (например, 1 или другое комплексное число), $R > 0$ (выбирайте поменьше: 10^{-3}, 10^{-4} и т.д) на комплексной плоскости рисует множества $P_i(R)$. Для этого необходимо перебрать (с небольшим шагом) всевозможные комплексные значения z_0, лежащие внутри видимой области. Для каждой точки необходимо запустить процесс вычисления z_n и проверки, не лежит ли текщий z_n на расстоянии меньшем R от корня уравнения. Когда это условие выполнится, то точка красится в цвет, соответствующий этому корню.

2. Используя программу, определите, как меняется изображение при изменении параметра c (помните, что он комплексный).

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая изображение бассейной Ньютона ($P_i(R)$), в зависимости от c и R.

2. Сделайте возможность сохранения построенной картинки в jpeg-файл.

3. Дайте возможность пользователю изменять параметры c (не забывайте, что он комплексный) и R.

4. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного рисунка.

5. Как будет изменяться изображение при изменении параметров?

Бассейны Ньютона - 2

Введение

Рассмотрим уравнение в комплексных числах

$$f(z) = z^4 + c = 0,$$

где c – комплексная постоянная. Метод Ньютона для нахождения решения такого уравнения состоит в последовательном вычислении z_n по правилу

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}$$

при некоторым начальном условии z_0. Предел последовательности – корень уравнения.

Выбор начального z_0 очень важен. Так как функция может иметь несколько нулей, то при различных z_0 метод может сходиться к различным корням. Однако, что за области обеспечат сходимость к тому или иному корню?

Бассейнами Ньютона назовем разбиение комплексной плоскости на множества, соответствующих одному и тому же корню $f(z) = 0$.

На практике, построение такого разбиения в точности невозможно, но можно построить его приближение. Обозначим через $P_i(R)$ множество таких комплексных чисел z_0, что в последовательности z_n найдется элемент, находящийся на расстоянии меньше R от i-го корня уравнения. Эти множества будут хорошим приближением настоящих бассейнов.

Задания

1. Напишите программу, которая при заданных значениях параметров c (например, 1 или другое комплексное число), $R > 0$ (выбирайте поменьше: 10^{-3}, 10^{-4} и т.д) на комплексной плоскости рисует множества $P_i(R)$. Для этого необходимо перебрать (с небольшим шагом) всевозможные комплексные значения z_0, лежащие внутри видимой области. Для каждой точки необходимо запустить процесс вычисления z_n и проверки, не лежит ли текщий z_n на расстоянии меньшем R от корня уравнения. Когда это условие выполнится, то точка красится в цвет, соответствующий этому корню.

2. Используя программу, определите, как меняется изображение при изменении параметра c (помните, что он комплексный).

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая изображение бассейной Ньютона ($P_i(R)$), в зависимости от c и R.

2. Сделайте возможность сохранения построенной картинки в jpeg-файл.

3. Дайте возможность пользователю изменять параметры c (не забывайте, что он комплексный) и R.

4. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного рисунка.

5. Как будет изменяться изображение при изменении параметров?

Бассейны Ньютона - 3

Введение

Рассмотрим уравнение в комплексных числах

$$f(z) = z^5 + c = 0,$$

где c – комплексная постоянная. Метод Ньютона для нахождения решения такого уравнения состоит в последовательном вычислении z_n по правилу

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}$$

при некоторым начальном условии z_0. Предел последовательности – корень уравнения.

Выбор начального z_0 очень важен. Так как функция может иметь несколько нулей, то при различных z_0 метод может сходиться к различным корням. Однако, что за области обеспечат сходимость к тому или иному корню?

Бассейнами Ньютона назовем разбиение комплексной плоскости на множества, соответствующих одному и тому же корню $f(z) = 0$.

На практике, построение такого разбиения в точности невозможно, но можно построить его приближение. Обозначим через $P_i(R)$ множество таких комплексных чисел z_0, что в последовательности z_n найдется элемент, находящийся на расстоянии меньше R от i-го корня уравнения. Эти множества будут хорошим приближением настоящих бассейнов.

Задания

1. Напишите программу, которая при заданных значениях параметров c (например, 1 или другое комплексное число), $R > 0$ (выбирайте поменьше: 10^{-3}, 10^{-4} и т.д) на комплексной плоскости рисует множества $P_i(R)$. Для этого необходимо перебрать (с небольшим шагом) всевозможные комплексные значения z_0, лежащие внутри видимой области. Для каждой точки необходимо запустить процесс вычисления z_n и проверки, не лежит ли текщий z_n на расстоянии меньшем R от корня уравнения. Когда это условие выполнится, то точка красится в цвет, соответствующий этому корню.

2. Используя программу, определите, как меняется изображение при изменении параметра c (помните, что он комплексный).

Ожидаемые результаты

1. Программа (исходный код и исполняемый файл), строящая изображение бассейной Ньютона ($P_i(R)$), в зависимости от c и R.

2. Сделайте возможность сохранения построенной картинки в jpeg-файл.

3. Дайте возможность пользователю изменять параметры c (не забывайте, что он комплексный) и R.

4. Сделайте возможность изменения масштаба по осям, сдвига координатной сетки, чтобы можно было детально рассмотреть любую часть построенного рисунка.

5. Как будет изменяться изображение при изменении параметров?