Домашнее задание №1

Материалы по этому заданию доступны по ссылке: https://github.com/v-biryukov/cs_mipt_faki/tree/master/term2
Для компиляции используйте компилятор g++ на Linux/MacOs или компилятор MinGW на Windows.

Класс Complex

В папке `classes/Complex` лежит реализация класса комплексных чисел. Подлючите файл `complex.h` к вашей программе и выполните следующее:

1. Создайте комплексное число $5 + 4i$, разделите его на i и напечатайте.
2. Создайте комплексное число $z = \frac{1+i}{\sqrt{2}}$, найдите чему равно z^8 и напечатайте.
3. Написать функцию `Complex sin(Complex z)`, которая вычисляет синус комплексного числа.
4. Создайте вектор комплесных чисел, заполните его 5-ю произвольными комплексными числами. Найдите сумму этих комплексных чисел двумя разными способами:
   (a) С помощью цикла `for`. Используйте итератор `std::vector<Complex>::iterator`.
   (b) С помощью функции `std::accumulate()` из библиотеки `numeric`.

Класс Image

В папке `classes/Image` лежит реализация класса изображений. Подлючите файл `image.h` к вашей программе.

Рисунок 1: Пример обработки изображения с помощью класса Image

1. **Компиляция:** Скомпилируйте файлы `image.cpp` и `main.cpp` в папке `classes/Image`. При компиляции вам нужно указать оба исполняемых файла: `g++ image.cpp main.cpp`. Запустите и посмотрите на результат работы программы.
2. **Black and White:** Добавьте метод `void blackwhite()` к классу `Image`, который будет делать изображение черно-белым. Подсказка: у черно-белого изображения все компоненты цвета равны.
3. ***Свёртка:** Добавьте метод `void convolve(int n, int m, float filter[])` к классу `Image`, который будет делать свёртку изображения с фильтром(другое название – ядро). `filter` это массив размера $n \times m$. Пиксель нового изображения с координатами x, y определяется следующим образом:

$$B[x][y] = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} A[x + i - n/2][y + j - m/2] * f[i][j]$$

, где A – старое изображение, B – новое, f – массив filter. При выходе за границы массива A берем в качестве значений нули.
По следующей ссылке можно найти примеры обработки изображения с помощью различных матричных фильтров https://en.wikipedia.org/wiki/Kernel_(image_processing) Добавьте методы `blur()`, `edges()` и `sharpen()` к классу `Image` которые будут вызывать метод `convolve` с различными фильтрами. Протестировать эти методы на различных изображениях.
Более подробно об операции свёртки: https://www.youtube.com/watch?v=X5O6wVmOYvk

4. **Задача об убегающей точке:** Предположим, что у нас есть комплексная функция $f(z) = z^2$. Выберем некоторое комплексное число z_0 и будем проводить следующие итерации: $z_1 = f(z_0), z_2 = f(z_1), ..., z_{n+1} = f(z_n)$. В зависимости от выбора точки z_0 эта последовательность либо разойдётся, либо останется в некоторой ограниченной области. Нужно найти все точки комплексной плоскости, которые не являются убегающими.
   Для функции $f(z) = z^2$ эта область тривиальна, но всё становится сложней для функции вида $f(z) = z^2 + c$, где c – некоторое комплексное число. Численно найдите область убегания для функций такого вида. Для этого создайте изображение размера 1000x1000, покрывающую область [-2:2]x[-2:2] на комплексной плоскости. Для каждой точки этой плоскости проведите $N = 50$ итераций и, в зависимости от результата, окрасьте пиксель в соответствующий цвет (цвет можно подобрать самим). Используйте классы Complex и Image. Программа должна создавать файл `juliaset.ppm`.
   Добавьте параметры командной строки: 2 вещественных числа, соответствующие комплексному числу c, и целое число итераций N.

5. **Множество:** Зафиксируем теперь $z_0 = 0$ и будем менять c. Численно найдите все параметры c, для которых точка не является убегающей. Программа должна создавать файл `mandelbset.ppm`.

STL