Паттерн Memento, Singleton, Bridge

Работа с изображениями

Реализовать паттерном Singleton, Bridge

- Класс изображение,
  В котором будут функции фильтров над этим изображением
  - функция Сглаживание изображения
  - функция Контрастность изображение
- Ссылку на интерфейс реализации, которая инициализируется с помощью конструктора класса изображения, интерфейс реализации является Singleton

Класс Изображение

Изображением считается матрица чисел(vector<vector<int>>)
хранящая код цвета каждого пикселя изображения
отсчет начинается с верхнего левого угла, а дальне нумерация идет слева -> направо и сверху -> вниз
Изображение должно уметь выводиться на экран, для этого можно воспользоваться следующим кодом, генерирующим html файл(сайт).

Класс изображение

А так же хранит в себе ссылку на интерфейс реализации(правило паттерна Bridge)
А также методы для фильтров.
Для применения любого фильтра, применяется понятие “Оператора преобразований”, который так же, как и изображение представляет собой матрицу, но более маленького размера, чем изображение

Пример матрицы: ”Оператора преобразования”		
1	6	2
9	-1	5
0	-7	3

Резкое изображение

Матрица данного преобразования выглядит следующим образом

-1	-1	-1
-1	9	-1
-1	-1	-1

Сглаженное изображение

Матрица данного преобразования выглядит следующим образом

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Как применять фильтры

Допустим, есть следующее изображение

Данное изображение будет выглядеть примерно так же, как результат кода из третьего слайда				
177	89	43	66	34
255	0	32	188	201
32	155	166	177	199
231	249	99	5	1
1	2	255	254	155

Для того, что применит любой фильтр, нужно для каждой ячейки матрицы, и её ближайших соседей выполнять сложение всех вышеперечисленных ячеек, но домноженных на соответствующие ячейки из матрицы преобразования фильтра

Пример на следующем слайде.

Пример для элемента из середины изображения, его значение 166

177	89	43	66	34
255	0	32	188	201
32	155	**166**	177	199
231	249	99	5	1
1	2	255	254	155

Матрица преобразования		
1	*0*	*1*
-1	***5***	*-1*
0	*1*	*0*

результат = 0 * *1* + 32 * *0* + 188 * *1* + 155 * *(-1)* + 166 * *5* + 177 * *(-1)* + 249 * *0* + 99 * *1* + 5 * *0* = 785

Результат вычислений записывает в копию матрицы тех же размером, под тем же индексом, если результат получается больше 255, то считать его равным 255, а если меньше 0, то равным 0.

Сама задача

Обеспечить пользователю работу с изображением, применение любых фильтров в любое время(Резкость и Сглаживание можно применять неограниченное количество раз, даже если они уже были применены)
А также обеспечить возможность создания снимков для сохранения первоначального или любого другого понравившегося варианта, к которому можно будет в любой момент вернуться.
Алгоритмы применения фильтров вынести в отдельный класс(паттерн Bridge)
|Вообще, каждый пиксель изображения представляет собой тройку цифр red = (255, 0, 0), black = (0, 0 , 0), purpur = (128, 0, 128) и т.д, для простоты считаем, что у наших цветов все 3 цифры одинаковые |

```cpp
#include <iostream>
#include <fstream>
#include <string>
#define  row    16
#define  column 16

using namespace std;
int main()
{
    ofstream foutHTML("index.html");
    for (int i = 0; i < row; ++i) {
        foutHTML << "<div>";
        for (int j = 0; j < column; ++j) {
            foutHTML << "<div style = \" display: inline-block; width: 10px; height: 10px; background-color: rgba(" +
                to_string(rand() % 255) + "," +
                to_string(rand() % 255) + "," +
                to_string(rand() % 255) + ", 1)\"></div>";
        }
        foutHTML << "</div>";
    }
    foutHTML.close();
    system("index.html");
}
```

ÐŸÐ¾Ð´ÑÐ²ÐµÑ‚Ð° Ð¿Ñ€Ð¾Ð³Ñ€Ð°Ð¼Ð¼Ð¾Ð¹ ÐºÐ¾Ð´Ð°