Лабораторная работа №4

«Попиксельная обработка изображений»

Ваша задача — написать набор фильтров для обработки изображений.
Цель: Научиться работать с двумерными структурами данных (представленными как массив пикселей), применять алгоритмы для модификации данных и получать визуальный результат своей работы.

Подготовительный этап: "Черный ящик"

В данной работе мы не будем изучать тонкости форматов хранения изображения. В данной работе вы будете использовать готовый класс для работы с изображениями. Ниже будет представлен код, который вам нужно перенести в файл класса вашей работы.

Ваш инструментарий:

- **Класс `Image`:**
  - `public Image(string filePath)` — конструктор, загружает картинку из файла.
  - `public void Save(string filePath)` — сохраняет измененную картинку в новый файл.
  - `public int Width` и `public int Height` — ширина и высота картинки в пикселях.
  - `public Pixel this[int x, int y]` — индексатор. Позволяет получить или изменить пиксель по координатам `x` и `y`. Например: `Pixel p = image[10, 20];`
- **Структура `Pixel`:**
  - `public byte R, G, B;` — три байта, хранящие красную, зеленую и синюю компоненты цвета (значения от 0 до 255).

Ваша задача — написать статические методы (фильтры), которые принимают на вход объект `Image` и изменяют его.

Задания:

Все методы должны находиться в одном классе

Все создаваемые методы должны быть расположены в отдельном статическом классе `Filters.cs` (создать самостоятельно).
Все создаваемые методы-фильтры должны быть статическими (`static`).

Задание 1. Негатив (Инверсия цвета)

Задача: Написать метод `public static void Invert(Image image)`, который инвертирует цвета на изображении.

Реализация:
Вам нужно пройтись вложенными циклами по всем пикселям изображения (от `x = 0` до `image.Width-1` и от `y = 0` до `image.Height-1`). Для каждого пикселя вы должны вычислить новый цвет по формуле:

- `newR = 255 - oldR`
- `newG = 255 - oldG`
- `newB = 255 - oldB`
  И записать новый пиксель обратно в изображение.

Демонстрация в `Main`:
Загрузите любую картинку, примените ваш фильтр и сохраните результат под новым именем. Сравните исходник и то, что получилось.

Задание 2. Оттенки серого (Grayscale)

Задача: Написать метод `public static void Grayscale(Image image)`, который делает изображение черно-белым.

Реализация:
Снова проходим по всем пикселям. Чтобы получить оттенок

серого, нужно "усреднить" цветовые компоненты. Но человеческий глаз воспринимает яркость цветов по-разному, поэтому простое среднее арифметическое (`(R+G+B)/3`) даст неточный результат. Используйте правильную, "взвешенную" формулу для вычисления яркости:

- `intensity = 0.3 * R + 0.59 * G + 0.11 * B`
  Новый цвет пикселя будет `newR = newG = newB = (byte)intensity` .

Задание 3. Сепия (эффект "под старину")

Задача: Написать метод `public static void Sepia(Image image)` , который придает изображению коричневатый оттенок старой фотографии.

Реализация:
Для каждого пикселя сначала вычисляется та же `intensity` , а затем новые компоненты цвета рассчитываются по специальным формулам:

- `newR = intensity + 2 * T`
- `newG = intensity + T`
- `newB = intensity`
  Где `T` — это глубина сепии, константа (например, `const int T = 20;`).
  Важно: Результаты могут выйти за пределы 255. Ваша задача - обрабатывать данную ошибку. (см. "Ограничение значений" в справочном материале).

Задание 4. Повышение яркости

Задача: Написать метод `public static void AdjustBrightness(Image image, int value)` , который делает изображение светлее или темнее.

Реализация:

1. **Публичный метод-оркестратор:** `public static void AdjustBrightness(Image image, int value)`
   - Этот метод отвечает за **проход по всем пикселям**. Он

содержит вложенные циклы `for`.
    - Внутри цикла нужно получить **прямую ссылку** на пиксель в памяти изображения с помощью специального метода `ref Pixel pixelRef = ref image.GetPixel(x, y);`.
    - После получения ссылки он вызывает второй метод, вспомогательный метод, передавая ему эту ссылку.
2. **Приватный метод-исполнитель:** `private static void AdjustPixelBrightness(ref Pixel p, int value)`
    - Этот метод **не знает** ни о каких циклах или изображениях. Его задача: взять **конкретный пиксель, переданный по ссылке**, и изменить его яркость.
    - Он применяет простую формулу: `newColor = oldColor + value` для каждого канала (R, G, B).
    - Здесь же он должен использовать механизм **ограничения значений**, чтобы результат не вышел за пределы диапазона 0-255.

Задание 5. Повышение контраста

Задача: Написать метод `public static void MakeContrast(Image image)`, который делает изображение светлее или темнее.

Реализация: Для каждого пикселя нужно вычислить новый цвет. Если цвет был темнее среднего (128), сделать его еще темнее. Если светлее — еще светлее. Формула:

Для каждого канала (R, G, B) `color`:

- `factor` = 1.5;
- `new_color` = 128 + (c - 128) * factor;

Задание 6. Простое размытие (Box Blur)

Задача: Написать метод `public static void BoxBlur(Image image)`, который размывает изображение.

Реализация: Здесь для вычисления нового цвета пикселя нужно учитывать не только его собственный цвет, но и цвета его соседей.

Реализация:

1. Создайте **копию** исходного изображения, чтобы читать из нее оригинальные цвета пикселей.
2. Пройдитесь по всем пикселям основного изображения (кроме самых крайних, чтобы не сталкиваться с границами).
3. Для каждого пикселя `(x, y)` посчитайте средний цвет его и 8 его соседей (квадрат 3x3). То есть, сложите все `R`, все `G` и все `B` из этого квадрата и разделите каждую сумму на 9.
4. Полученный усредненный цвет запишите в пиксель `(x, y)` **основного** изображения.

Задание 7. Корректировка определенного цвета

Задача: Написать метод `public static void AdjustChannels(Image image, sbyte deltaR, sbyte deltaG, sbyte deltaB)`, который позволяет выборочно усилить или ослабить каждый цветовой канал.

Реализация:

1. **Сигнатура:** `public static void AdjustChannels(Image image, sbyte deltaR, sbyte deltaG, sbyte deltaB)`. На вход приходят три **знаковых байта**
2. **Алгоритм:**
   - Для каждого пикселя `p`:
     - `int newR = p.R + deltaR;`
     - `int newG = p.G + deltaG;`
     - `int newB = p.B + deltaB;`
   - Ограничиваем `newR`, `newG`, `newB` диапазоном 0-255.
   - Записываем обратно в пиксель.

Пример использования

```
// 1. Загружаем изображение. Убедитесь, что файл "input.bmp" лежит в папке с exe.
Console.WriteLine("Загрузка изображения 'input.bmp'...");
Image image = new Image("blue-bmp-24-bit.bmp");
// добавить сюда пример с Абсолютным путем картинки
Console.WriteLine($"Изображение загружено. Размеры:
```

```
{image.Width}x{image.Height}");

// 2. Применяем какой-нибудь фильтр (например, из вашей будущей библиотеки)
Console.WriteLine("Применяем фильтр инверсии...");
ImageFilters.Invert(image); // Предполагаем, что такой метод у вас есть

// 3. Сохраняем результат в новый файл.
string outputFileName = "output_inverted.bmp";
Console.WriteLine($"Сохранение результата в '{outputFileName}'...");
image.Save(outputFileName);

Console.WriteLine("Готово! Проверьте файл в папке с программой.");
```

Вариативность:

```
string filterName = "grayscale"; // Или считать с консоли/аргументов командной строки

switch (filterName)
{
    case "grayscale":
        Console.WriteLine("Применяем фильтр оттенков серого...");
        ImageFilters.Grayscale(image);
        break;
    case "sepia":
        Console.WriteLine("Применяем сепию...");
        ImageFilters.Sepia(image);
        break;
    default:
        Console.WriteLine("Неизвестный фильтр. Пропускаем этап фильтрации.");
        break;
}
```