

# Лабораторна робота №4

<b>Тема</b>	<b>1</b>
<b>Мета</b>	<b>1</b>
<b>Завдання</b>	<b>1</b>
Аргументи командного рядка	1
Вимоги до структури коду	2
<b>Методичні вказівки</b>	<b>2</b>
Модуль обробки аргументів командного рядка	3
Модулі редагування зображень	3
<b>Звіт</b>	<b>4</b>
Приклад частини звіту	4
Hue	4
RotateRight90	5
<b>Контрольні питання</b>	<b>6</b>
<b>Додатки</b>	<b>6</b>
Додаток А. Варіант методів редагування зображень	6

## Тема

Бібліотеки і обробка зображень

## Мета

Реалізувати різні алгоритми редагування зображень.

Розбити проект програми на декілька проектів у одному рішенні з використанням бібліотек класів.

## Завдання

Створити консольну програму, що дозволяє виконувати редагування зображень.

## Аргументи командного рядка

Приклад аргументів:

``dotnet run {module} ./file.jpg ./out.jpg`` - аргументи обов'язкові і зберігають такий порядок, тільки цих аргументів недостатньо, після них задавати команду редагування і її параметри

- `{module}` - `pixel` або `fast`, визначає яким саме модулем редагування змінити зображення.
- `./file.jpg` - перший аргумент після `{module}` - приклад шляху вхідного зображення
- `./out.jpg` - другий аргумент після `{module}` - приклад шляху вихідного зображення

Команди:

- Команда отримання частини зображення за координатами:  
`crop {width}x{height}+{left}+{top}` - всі аргументи обов'язкові і зберігають такий порядок.  
 Приклад: `dotnet run pixel ./file.jpg ./out.jpg crop 100x100+30+90`
- Команди методів за варіантом із таблиць 1-4 (див. **Додаток А**).

## Вимоги до структури коду

Розбити програму на модулі:

- **модуль обробки аргументів командного рядка** - модуль аналізує задані користувачем аргументи командного рядка і використовує інші модулі.
- **модулі редагування зображень** - містять функції, що на основі вхідного зображення створюють змінене зображення.
  - реалізація за допомогою стандартних функцій, матриць кольору, матриць трансформації або будь-якої графічної бібліотеки
  - реалізація за допомогою піксельних змін

Створити бібліотеки:

- **ProgbaseLab.ImageEditor.Common** - містить контракт модулів редагування
- **ProgbaseLab.ImageEditor.Pixel** - містить модуль редагування зображень пікселями
- **ProgbaseLab.ImageEditor.Fast** - містить модуль редагування зображень стандартними функціями або з використанням інших графічних бібліотек

Підключити і використати бібліотеки у проекті консольної програми.

## Методичні вказівки

Рекомендовані кроки виконання:

1. Створити консольний проект.
  - a. Реалізувати модуль обробки аргументів командного рядка
  - b. Реалізувати один із модулів редагування зображень
2. Створити рішення (solution) і додати у нього консольний проект
3. Винести код модуля редагування зображень у новий проект бібліотеки класів, додати проект у рішення і підключити бібліотеку до проекту консольної програми
4. Додати контракт модулів редагування зображення і винести її у власну бібліотеку. Підключити цю бібліотеку до обох проектів і використати у них контракт

5. Реалізувати другий модуль редагування зображення у окремій бібліотеці. Підключити до неї бібліотеку з контрактом і підключити та використати цю бібліотеку у консольному проекті

## Модуль обробки аргументів командного рядка

На основі аргументів командного рядка виконати команду із відповідного модуля.

Обробку кожної команди виконувати у окремій функції.

Після виконання команди вивести у консоль час її виконання і окремо час виконання тільки виклику функції, що виконує редагування зображення (використати для заміру часу

**Diagnostics.Stopwatch**) і завершити програму з кодом 0.

Якщо команду виконати неможливо, вивести у потік помилок (**Console.Error**) повідомлення і завершити програму з кодом 1.

## Модулі редагування зображень

Перед тим як реалізувати два модуля редагування зображень і їх контракт, можна перший модуль зробити за допомогою статичного класу, а потім переписати на клас, що реалізує контракт редагування.

Приклад статичного класу для модуля (використати методи за варіантом):

```
namespace ProgbaseLab.ImageEditorLib
{
    public static class ImageEditor
    {
        public static Bitmap Crop(Bitmap bmp, int left, int top, int width, int height)
        {
            throw new System.NotImplementedException();
        }

        public static Bitmap RotateRight90(Bitmap bmp)
        {
            throw new System.NotImplementedException();
        }

        public static Bitmap ChangeSaturation(Bitmap bmp, int saturation)
        {
            throw new System.NotImplementedException();
        }
    }
}
```

Приклад контракту для модулів редагування (на основі попереднього прикладу):

```
namespace ProgbaseLab.ImageEditorLib
{
    public interface IImageEditor
    {
        Bitmap Crop(Bitmap bmp, int left, int top, int width, int height);
    }
}
```

```
Bitmap RotateRight90(Bitmap bmp);  
  
Bitmap ChangeSaturation(Bitmap bmp, int saturation);  
}  
}
```

Поточний модуль редагування, що буде використовуватись у програмі можна задавати аргументом командного рядка.

## Звіт

Обрати кольорове зображення для демонстрації роботи програми. Зображення не має бути квадратним і має мати насичені кольори (відтінки червоного, зеленого і синього).



У звіт додати оригінальне зображення до змін, зображення після змін через розроблену програму за допомогою кожного з двох реалізованих модулів і після таких же змін оригінального зображення за допомогою будь-якого існуючого редактора зображень (Paint, Paint.net, Pinta, Gimp, Photoshop або ін.).

Для різних реалізацій обов'язково додати час виконання операції.

## Приклад частини звіту

### Нue

Оригінал і очікуваний результат:



Оригінал	Paint.net
	

Порівняння реалізацій:

```
dotnet run pixel ./file.jpg ./out.jpg hue 100
```


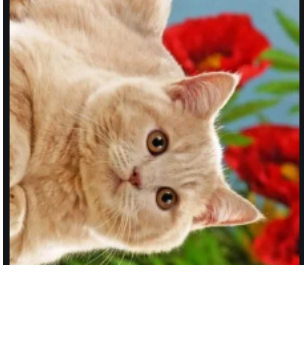
33 ms
<code>dotnet run fast ./file.jpg ./out.jpg hue 100</code>
3 ms

Результати:

ImageEditor.Pixel (піксельна реалізація)	ImageEditor.Fast (швидка реалізація)
	

## RotateRight90

Оригінал і очікуваний результат:

Оригінал	Paint.net
	

Порівняння реалізацій:

<code>dotnet run pixel ./file.jpg ./out.jpg rotate-right-90</code>
3 ms
<code>dotnet run fast ./file.jpg ./out.jpg rotate-right-90</code>
1 ms

Результати:

ImageEditor.Pixel (піксельна реалізація)	ImageEditor.Fast (швидка реалізація)
	

## Контрольні питання

Скоро будуть.

## Додатки

### Додаток А. Варіант методів редагування зображень

n - номер у списку групи.

Таблиця 1. Варіант першого метода

n % 5	Метод	Опис
0	RotateRight90	Повернути зображення на 90 градусів за годинниковою стрілкою
1	RotateLeft90	Повернути зображення на 90 градусів проти годинникової стрілки
2	Rotate180	Повернути зображення на 180 градусів
3	FlipVertical	Віддзеркалити зображення вертикально
4	FlipHorizontal	Віддзеркалити зображення горизонтально

Таблиця 2. Варіант другого метода

n % 9	Метод	Опис
0	SwapRedAndGreen	Поміняти значення червоного і зеленого каналів
1	SwapRedAndBlue	Поміняти значення червоного і синього каналів
2	SwapGreenAndBlue	Поміняти значення зеленого і синього каналів
3	RemoveRed	Обнулити червоний канал кольору
4	RemoveGreen	Обнулити зелений канал кольору

5	RemoveBlue	Обнулити синій канал кольору
6	ExtractRed	Залишити тільки червоний канал кольору, інші обнулити
7	ExtractGreen	Залишити тільки зелений канал кольору, інші обнулити
8	ExtractBlue	Залишити тільки синій канал кольору, інші обнулити

*Таблиця 3. Варіант третього метода*

n % 3	Метод	Опис
0	Grayscale	Зробити зображення у відтінках сірого
1	Sepia	Застосувати ефект сепії
2	InvertColors	Інвертувати кольори

*Таблиця 4. Варіант четвертого метода*

n % 5	Метод	Параметри
0	ChangeHue	hue - від -180 до 180 градусів
1	ChangeSaturation	saturation - від 0 до 200
2	ChangeBrightness	brightness - від -100 до 100
3	Blur	sigma - від 0 до 20. <i>Gaussian Blur</i>
4	Sharpen	amount - від 1 до 20