

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 4

з дисципліни “Основи програмування”

# тема “Бібліотеки і обробка зображень”

|  |  |  |
| --- | --- | --- |
| Виконав(ла)  студент(ка) I курсу  групи КП-02  Жученко Андрій Сергійович  (*прізвище, ім’я, по батькові*) |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2020

# 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 A**).

## 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.

## 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.

## Варіант методів редагування зображень

**n** - 4; номер у списку групи.

*Таблиця 1. Варіант першого метода*

|  |  |  |
| --- | --- | --- |
| **n % 5** | **Метод** | **Опис** |
| 4 | FlipHorizontal | Віддзеркалити зображення горизонтально |

*Таблиця 2. Варіант другого метода*

|  |  |  |
| --- | --- | --- |
| **n % 9** | **Метод** | **Опис** |
| 4 | RemoveGreen | Обнулити зелений канал кольору |

*Таблиця 3. Варіант третього метода*

|  |  |  |
| --- | --- | --- |
| **n % 3** | **Метод** | **Опис** |
| 1 | Sepia | Застосувати ефект сепії |

*Таблиця 4. Варіант четвертого метода*

|  |  |  |
| --- | --- | --- |
| **n % 5** | **Метод** | **Параметри** |
| 4 | Sharpen | amount - від 1 до 20 |

**Тексти коду програм**

|  |
| --- |
| **./App/Program.cs** |
| using System;  using System.Drawing;  using OpenCvSharp;  using System.Diagnostics;  namespace lab4\_2  {  class Program  {  static void Main(string[] args)  {  Mat bmp = new Mat("./inputImage.png");  CheckArguments(args);  if(args[3] == "sharpen")  {  ProcessCommandSharpen(args);  }  else if(args[3] == "removeGreen")  {  ProcessCommandRemoveGreen(args);  }  else if(args[3] == "sepia")  {  ProcessCommandSepia(args);  }  else if(args[3] == "crop")  {  ProcessCommandCrop(args);  }  else if(args[3] == "flipHorizontal")  {  ProcessCommandFlipHorizontal(args);  }  }  static void CheckArguments(string[] args)  {  if(args.Length < 4)  {  throw new ArgumentException("Not enough aruments.");  }  if(args[0] != "pixel" && args[0] != "fast")  {  throw new ArgumentException("Wrong name of module.");  }  try  {  Bitmap check = new Bitmap(args[1]);  }  catch  {  throw new ArgumentException("File does not exit or file is not an image.");  }  if(args[3] != "crop" && args[3] != "flipHorizontal" && args[3] != "removeGreen" && args[3] != "sepia" && args[3] != "sharpen")  {  throw new ArgumentException("Wrong method.");  }  if(args[3] == "crop" && args.Length <5)  {  throw new ArgumentException("For method 'crop' you should input additional arguments.");  }  if(args[3] == "sharpen" && args.Length < 5)  {  throw new ArgumentException("For method 'sharpen' you should input additional arguments.");  }  if((args.Length > 4 && (args[3] != "sharpen" && args[3] != "crop")) || ((args[3] == "sharpen" || args[3] == "crop") && args.Length > 5))  {  throw new ArgumentException("Too many arguments.");  }  if(args[3] == "sharpen")  {  try  {  int checkSharpenArgument = int.Parse(args[4]);  }  catch  {  throw new ArgumentException("Value for 'sharpen' should be integer.");  }  finally  {  if(int.Parse(args[4]) > 20 || int.Parse(args[4]) < 1)  {  throw new ArgumentException("Value for 'sharpen' should be in range of 1 to 20.");  }  }  }  if(args[3] == "crop")  {  int[] cropValues = new int[4];  string[] temporary = args[4].Split('+');  if(temporary.Length != 3)  {  throw new Exception("Wrong additional value for method 'crop'");  }  string[] temporary2 = temporary[0].Split('x');  if(temporary2.Length != 2)  {  throw new Exception("Wrong additional value for method 'crop'");  }  try  {  cropValues[0] = int.Parse(temporary2[0]);  cropValues[1] = int.Parse(temporary2[1]);  cropValues[2] = int.Parse(temporary[1]);  cropValues[3] = int.Parse(temporary[2]);  }  catch  {  throw new Exception("Wrong additional value for method'crop'");  }  finally  {  Bitmap image = new Bitmap(args[1]);  if(int.Parse(temporary[2]) + int.Parse(temporary2[1]) > image.Height)  {  throw new Exception("Inputted rectangke cannot be cropped.");  }  if(int.Parse(temporary[1]) + int.Parse(temporary2[0]) > image.Width)  {  throw new Exception("Inputted rectangke cannot be cropped.");  }  }  }  }  static void ProcessCommandCrop(string[] args)  {  int[] cropValues = new int[4];  string[] temporary = args[4].Split('+');  string[] temporary2 = temporary[0].Split('x');  int left = int.Parse(temporary[1]);  int top = int.Parse(temporary[2]);  int width = int.Parse(temporary2[0]);  int height = int.Parse(temporary2[1]);  if(args[0] == "fast")  {  Bitmap image = new Bitmap(args[1]);  ProgbaseLab.ImageEditor.Fast.FastImageEditor imageEditor = new ProgbaseLab.ImageEditor.Fast.FastImageEditor();  Stopwatch stopWatch = new Stopwatch();  stopWatch.Start();  Bitmap result = imageEditor.Crop(image, left, top, width, height);  stopWatch.Stop();  TimeSpan ts = stopWatch.Elapsed;  Console.WriteLine($"Time:{ts}");  result.Save(args[2]);  }  else  {  Bitmap image = new Bitmap(args[1]);  ProgbaseLab.ImageEditor.Pixel.PixelImageEditor imageEditor = new ProgbaseLab.ImageEditor.Pixel.PixelImageEditor();  Stopwatch stopWatch = new Stopwatch();  stopWatch.Start();  Bitmap result = imageEditor.Crop(image, left, top, width, height);  stopWatch.Stop();  TimeSpan ts = stopWatch.Elapsed;  Console.WriteLine($"Time:{ts}");  result.Save(args[2]);  }  }  static void ProcessCommandFlipHorizontal(string[] args)  {  if(args[0] == "fast")  {  Bitmap image = new Bitmap(args[1]);  ProgbaseLab.ImageEditor.Fast.FastImageEditor imageEditor = new ProgbaseLab.ImageEditor.Fast.FastImageEditor();  Stopwatch stopWatch = new Stopwatch();  stopWatch.Start();  Bitmap result = imageEditor.FlipHorizontal(image);  stopWatch.Stop();  TimeSpan ts = stopWatch.Elapsed;  Console.WriteLine($"Time:{ts}");  result.Save(args[2]);  }  else  {  Bitmap image = new Bitmap(args[1]);  ProgbaseLab.ImageEditor.Pixel.PixelImageEditor imageEditor = new ProgbaseLab.ImageEditor.Pixel.PixelImageEditor();  Stopwatch stopWatch = new Stopwatch();  stopWatch.Start();  Bitmap result = imageEditor.FlipHorizontal(image);  stopWatch.Stop();  TimeSpan ts = stopWatch.Elapsed;  Console.WriteLine($"Time:{ts}");  result.Save(args[2]);  }  }  static void ProcessCommandSharpen(string[] args)  {  if(args[0] == "fast")  {  Bitmap image = new Bitmap(args[1]);  int amount = int.Parse(args[4]);  ProgbaseLab.ImageEditor.Fast.FastImageEditor imageEditor = new ProgbaseLab.ImageEditor.Fast.FastImageEditor();  Stopwatch stopWatch = new Stopwatch();  stopWatch.Start();  Bitmap result = imageEditor.Sharpen(image, amount);  stopWatch.Stop();  TimeSpan ts = stopWatch.Elapsed;  Console.WriteLine($"Time:{ts}");  result.Save(args[2]);  }  else  {  Bitmap image = new Bitmap(args[1]);  int amount = int.Parse(args[4]);  ProgbaseLab.ImageEditor.Pixel.PixelImageEditor imageEditor = new ProgbaseLab.ImageEditor.Pixel.PixelImageEditor();  Stopwatch stopWatch = new Stopwatch();  stopWatch.Start();  Bitmap result = imageEditor.Sharpen(image, amount);  stopWatch.Stop();  TimeSpan ts = stopWatch.Elapsed;  Console.WriteLine($"Time:{ts}");  result.Save(args[2]);  }  }  static void ProcessCommandRemoveGreen(string[] args)  {  if(args[0] == "fast")  {  Bitmap image = new Bitmap(args[1]);  ProgbaseLab.ImageEditor.Fast.FastImageEditor imageEditor = new ProgbaseLab.ImageEditor.Fast.FastImageEditor();  Stopwatch stopWatch = new Stopwatch();  stopWatch.Start();  Bitmap result = imageEditor.RemoveGreen(image);  stopWatch.Stop();  TimeSpan ts = stopWatch.Elapsed;  Console.WriteLine($"Time:{ts}");  result.Save(args[2]);  }  else  {  Bitmap image = new Bitmap(args[1]);  ProgbaseLab.ImageEditor.Pixel.PixelImageEditor imageEditor = new ProgbaseLab.ImageEditor.Pixel.PixelImageEditor();  Stopwatch stopWatch = new Stopwatch();  stopWatch.Start();  Bitmap result = imageEditor.RemoveGreen(image);  stopWatch.Stop();  TimeSpan ts = stopWatch.Elapsed;  Console.WriteLine($"Time:{ts}");  result.Save(args[2]);  }  }  static void ProcessCommandSepia(string[] args)  {  if(args[0] == "fast")  {  Bitmap image = new Bitmap(args[1]);  ProgbaseLab.ImageEditor.Fast.FastImageEditor imageEditor = new ProgbaseLab.ImageEditor.Fast.FastImageEditor();  Stopwatch stopWatch = new Stopwatch();  stopWatch.Start();  Bitmap result = imageEditor.Sepia(image);  stopWatch.Stop();  TimeSpan ts = stopWatch.Elapsed;  Console.WriteLine($"Time:{ts}");  result.Save(args[2]);  }  else  {  Bitmap image = new Bitmap(args[1]);  ProgbaseLab.ImageEditor.Pixel.PixelImageEditor imageEditor = new ProgbaseLab.ImageEditor.Pixel.PixelImageEditor();  Stopwatch stopWatch = new Stopwatch();  stopWatch.Start();  Bitmap result = imageEditor.Sepia(image);  stopWatch.Stop();  TimeSpan ts = stopWatch.Elapsed;  Console.WriteLine($"Time:{ts}");  result.Save(args[2]);  }  }  }  } |

|  |
| --- |
| **./ProgbaseLab.ImageEditor.Common/IImageEditor.cs** |
| using System.Drawing;  namespace ProgbaseLab.ImageEditor.Common  {  public interface IImageEditor  {  Bitmap Crop(Bitmap bmp, int left, int top, int width, int height);  Bitmap FlipHorizontal(Bitmap image);  Bitmap RemoveGreen(Bitmap image);  Bitmap Sepia(Bitmap image);  Bitmap Sharpen(Bitmap image, int amount);  }  } |

|  |
| --- |
| **./ProgbaseLab.ImageEditor.Fast/FastImageEditorClass.cs** |
| using System;  using System.Drawing;  using OpenCvSharp;  using OpenCvSharp.Extensions;  using System.Drawing.Imaging;  namespace ProgbaseLab.ImageEditor.Fast  {  public class FastImageEditor : ImageEditor.Common.IImageEditor  {  public Bitmap Crop(Bitmap bmp, int left, int top, int width, int height)  {  Mat mat = BitmapConverter.ToMat(bmp);  Rect rect = new Rect(left, top, width, height);  Mat croppedImage = new Mat(mat, rect);  return BitmapConverter.ToBitmap(croppedImage);  }  public Bitmap FlipHorizontal(Bitmap image)  {  image.RotateFlip(RotateFlipType.RotateNoneFlipX);  return image;  }  public Bitmap RemoveGreen(Bitmap image)  {  Mat bmp = BitmapConverter.ToMat(image);  Mat[] channels = Cv2.Split(bmp);  channels[1].SetTo(0);  Mat result = new Mat();  Cv2.Merge(channels, result);  return BitmapConverter.ToBitmap(result);  }  public Bitmap Sepia(Bitmap image)  {  Bitmap result = new Bitmap(image.Width, image.Height);  Graphics g = Graphics.FromImage(result);  ColorMatrix colorMatrix = SepiaMatrix();  ImageAttributes attributes = new ImageAttributes();  attributes.SetColorMatrix(colorMatrix);  g.DrawImage(image, new Rectangle(0, 0 ,image.Width, image.Height), 0, 0, image.Width, image.Height, GraphicsUnit.Pixel, attributes);  attributes.Dispose();  g.Dispose();  return result;  }  public Bitmap Sharpen(Bitmap image, int amount)  {  Mat bmp = BitmapConverter.ToMat(image);  Mat result = new Mat();  float[,] data = {  {-1, -1, -1, },  {-1, 9, -1, },  {-1, -1, -1, },  };  for(int i = 0; i < 3; i++)  {  for(int j = 0; j < 3; j++)  {  data[i,j] = data[i, j] \* amount / 2;  }  }  Mat kernel = new Mat(data.GetLength(0), data.GetLength(1), MatType.CV\_32FC1, data);  Cv2.Filter2D(bmp, result, -1, kernel);  Bitmap convertedResult = BitmapConverter.ToBitmap(result);  return convertedResult;  }  private static ColorMatrix SepiaMatrix()  {  return new ColorMatrix(  new float[][]  {  new float[] {0.393f, 0.349f, 0.272f, 0, 0},  new float[] {0.769f, 0.686f, 0.534f, 0, 0},  new float[] {0.189f, 0.168f, 0.131f, 0, 0},  new float[] {0, 0, 0, 1, 0},  new float[] {0, 0, 0, 0, 1}  });  }  }  } |

|  |
| --- |
| **./ProgbaseLab.ImageEditor.Pixel/PixelImageEditorClass.cs** |
| using System;  using System.Drawing;  namespace ProgbaseLab.ImageEditor.Pixel  {  public class PixelImageEditor : ImageEditor.Common.IImageEditor  {  public Bitmap Crop(Bitmap bmp, int left, int top, int width, int height)  {  Bitmap result = new Bitmap(width, height);  int counter1 = 0;  for(int i = top; i < top + height; i ++)  {  int counter2 = 0;  for(int j = left; j < left + width; j++)  {  result.SetPixel(counter2, counter1, bmp.GetPixel(j, i));  counter2++;  }  counter1++;  }  return result;  }  public Bitmap FlipHorizontal(Bitmap image)  {  Bitmap result = new Bitmap(image.Width, image.Height);  for(int i = 0; i < image.Width / 2; i++)  {  int counter = image.Height;  for(int j = 0; j < image.Height; j++)  {  result.SetPixel(i, counter - 1, image.GetPixel(i, j));  counter--;  }  }  for(int i = image.Width / 2; i < image.Width; i++)  {  int counter = image.Height;  for(int j = 0; j < image.Height; j++)  {  result.SetPixel(i, counter - 1, image.GetPixel(i, j));  counter--;  }  }  result.RotateFlip(RotateFlipType.Rotate180FlipNone);  return result;  }  public Bitmap RemoveGreen(Bitmap image)  {  for(int i = 0; i < image.Height; i++)  {  for (int j = 0; j < image.Width; j++)  {  Color color = image.GetPixel(j, i);  Color newColor = Color.FromArgb(color.A, color.R, 0, color.B);  image.SetPixel(j, i, newColor);  }  }  return image;  }  public Bitmap Sepia(Bitmap image)  {  Bitmap resut = new Bitmap(image.Width, image.Height);  for(int i = 0; i < image.Height; i++)  {  for(int j = 0; j < image.Width; j++)  {  Color color = image.GetPixel(j, i);  int R = color.R;  int G = color.G;  int B = color.B;  int newRed = (int)Math.Min(255, 0.393\*R + 0.769\*G + 0.189\*B);  int newGreen = (int)Math.Min(255, 0.349\*R + 0.686\*G + 0.168\*B);  int newBlue = (int)Math.Min(255, 0.272\*R + 0.534\*G + 0.131\*B);  Color newColor = Color.FromArgb(255, newRed, newGreen, newBlue);  resut.SetPixel(j, i, newColor);  }  }  return resut;  }  public Bitmap Sharpen(Bitmap image, int amount)  {  Bitmap result = new Bitmap(image.Width, image.Height);  double[,] filter = new double[,] {  {-1, -1, -1},  {-1, 9, -1},  {-1, -1, -1}  };  for(int x = 0; x < image.Width; x++)  {  for(int y = 0; y < image.Height; y++)  {  Color newColor = ApplySharpFilter(image, x, y, filter, amount);  result.SetPixel(x, y, newColor);  }  }  return result;  }  private static Color ApplySharpFilter(Bitmap image, int x, int y, double[,] filter, double factor)  {  double red = 0.0;  double green = 0.0;  double blue = 0.0;  int filterSize = filter.GetLength(0);  int radius = filterSize / 2;  int w = image.Width;  int h = image.Height;  for (int filterX = -radius; filterX <= radius; filterX++)  {  for (int filterY = -radius; filterY <= radius; filterY++)  {  double filterValue = filter[filterX + radius, filterY + radius];  int imageX = (x + filterX + w) % w;  int imageY = (y + filterY + h) % h;  Color imageColor = image.GetPixel(imageX, imageY);  red += imageColor.R \* filterValue;  green += imageColor.G \* filterValue;  blue += imageColor.B \* filterValue;  }  }  int r = Math.Min(Math.Max((int)(factor \* red), 0), 255);  int g = Math.Min(Math.Max((int)(factor \* green), 0), 255);  int b = Math.Min(Math.Max((int)(factor \* blue), 0), 255);  return Color.FromArgb(r, g, b);  }  }  } |

# Результати роботи програми

Оригінал:

### 

### Crop

Порівняння реалізацій:

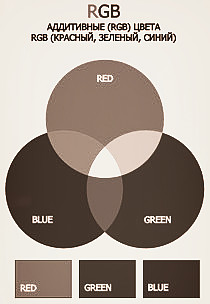
|  |
| --- |
| dotnet run pixel ./inputImage.png ./out.jpg crop 100x100+40+100 |
| Time:00:00:00.0106882 |
| dotnet run fast ./inputImage.png ./out.jpg crop 100x100+40+100 |
| Time:00:00:00.0196700 |

Результати:

|  |  |
| --- | --- |
| **ImageEditor.Pixel (піксельна реалізація)** | **ImageEditor.Fast (швидка реалізація)** |
|  |  |

### Sepia

Очікуваний результат:



Порівняння реалізацій:

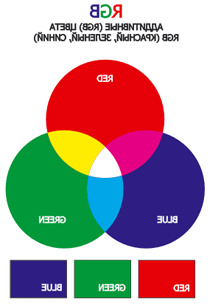
|  |
| --- |
| dotnet run pixel ./inputImage.png ./out.jpg sepia |
| Time:00:00:00.0337853 |
| dotnet run fast ./inputImage.png ./out.jpg sepia |
| Time:00:00:00.0536753 |

Результати:

|  |  |
| --- | --- |
| **ImageEditor.Pixel (піксельна реалізація)** | **ImageEditor.Fast (швидка реалізація)** |
|  |  |

### FlipHorizontal

Очікуваний результат:



Порівняння реалізацій:

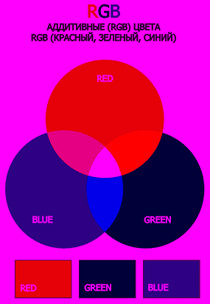
|  |
| --- |
| dotnet run pixel ./inputImage.png ./out.jpg flipHorizontal |
| Time:00:00:00.0312522 |
| dotnet run fast ./inputImage.png ./out.jpg flipHorizontal |
| Time:00:00:00.0027778 |

Результати:

|  |  |
| --- | --- |
| **ImageEditor.Pixel (піксельна реалізація)** | **ImageEditor.Fast (швидка реалізація)** |
|  |  |

### RemoveGreen

Очікуваний результат:



Порівняння реалізацій:

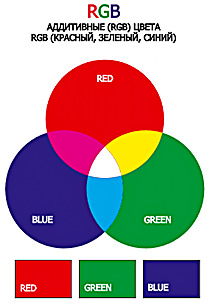
|  |
| --- |
| dotnet run pixel ./inputImage.png ./out.jpg removeGreen |
| Time:00:00:00.0284494 |
| dotnet run fast ./inputImage.png ./out.jpg removeGreen |
| Time:00:00:00.0393084 |

Результати:

|  |  |
| --- | --- |
| **ImageEditor.Pixel (піксельна реалізація)** | **ImageEditor.Fast (швидка реалізація)** |
|  |  |

### RemoveGreen

Очікуваний результат:



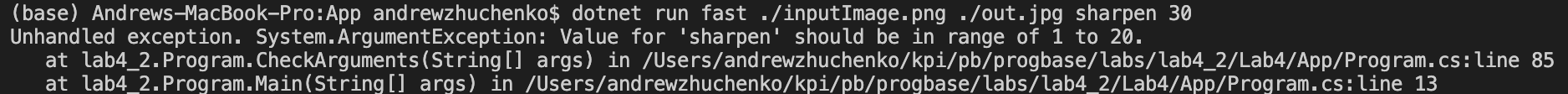
Порівняння реалізацій:

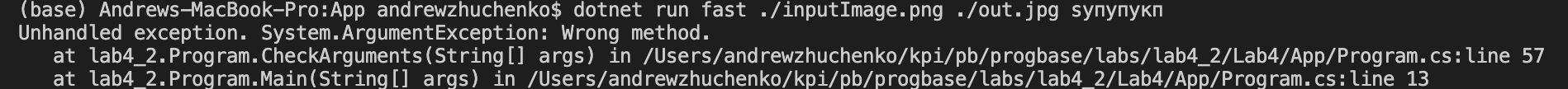
|  |
| --- |
| dotnet run pixel ./inputImage.png ./out.jpg sharpen 1 |
| Time:00:00:00.1044270 |
| dotnet run fast ./inputImage.png ./out.jpg sharpen 1 |
| Time:00:00:00.0238874 |

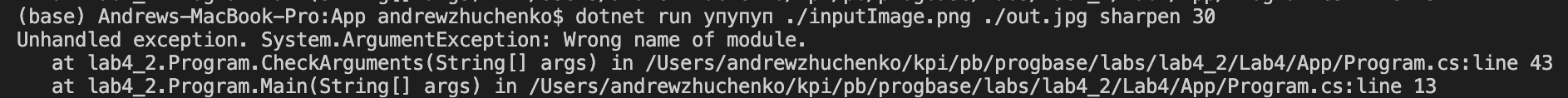
Результати:

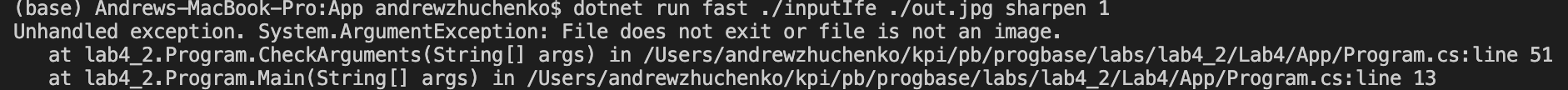
|  |  |
| --- | --- |
| **ImageEditor.Pixel (піксельна реалізація)** | **ImageEditor.Fast (швидка реалізація)** |
|  |  |

Приклади роботи програми при неправильних командах:









# Висновки

В результаті виконання цієї лабораторної роботи були реалізовані різні алгоритми редагування зображень. Проект програми був розбитий на декілька проектів у одному рішенні з використанням бібліотек класів. Створена консольна програма, що дозволяє виконувати редагування зображень.