НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ ТА ПРОГРАМУВАННЯ»

«Комп'ютерна графіка»

Звіт з лабораторної роботи №8

Тема: «РОБОТА З РАСТРОВИМИ ЗОБРАЖЕННЯМИ»

Виконав:

Студент групи КІТ-119а

Момот Р.Є.

Харків – 2021

Лабораторна робота №8

РОБОТА З РАСТРОВИМИ ЗОБРАЖЕННЯМИ

Ціль: вивчити можливості Visual Studio з відкриття та збереження файлів. Написати і налагодити програму для створення або обробки зображень.

Індивідуальне завдання: Додайте до наведеного графічного редактору свої функції відповідно до варіанту. Розширте додаток шляхом додавання можливості регулювання яскравості і контрастності як відразу за трьома каналами, так і по кожному каналу окремо.

1. Створіть функцію, розбиває зображення на чотири рівні частини. У кожній залиште значення тільки одного каналу R, G і B, а в четвертій виведіть градації сірого кольору.

Результати роботи

Код програми

Файл ColorBalance.cs

```
using System;

namespace Program
{
   class ColorBalance
   {
      //цветовой баланс R
      public static UInt32 ColorBalance_R(UInt32 point, int poz, int lenght)
      {
         int R;
         int G;
         int B;
```

```
    int N = (100 / lenght) * poz; //кол-во процентов

    R = (int)(((point & 0x00FF0000) >> 16) + N * 128 / 100);
    G = (int)((point & 0x0000FF00) >> 8);
    B = (int)(point & 0x000000FF);

    //контролируем переполнение переменных
    if (R < 0) R = 0;
    if (R > 255) R = 255;

    point = 0xFF000000 | ((UInt32)R << 16) | ((UInt32)G << 8) | ((UInt32)B);

    return point;
}

//цветовой баланс G
public static UInt32 ColorBalance_G(UInt32 point, int poz, int lenght)
{
    int R;
    int G;
    int B;

    int N = (100 / lenght) * poz; //кол-во процентов

    R = (int)((point & 0x00FF0000) >> 16);
    G = (int)(((point & 0x0000FF00) >> 8) + N * 128 / 100);
    B = (int)(point & 0x000000FF);

    //контролируем переполнение переменных
    if (G < 0) G = 0;
    if (G > 255) G = 255;
```

```
            point = 0xFF000000 | ((UInt32)R << 16) | ((UInt32)G << 8) | ((UInt32)B);

            return point;
        }

        //цветовой баланс B
        public static UInt32 ColorBalance_B(UInt32 point, int poz, int lenght)
        {
            int R;
            int G;
            int B;

            int N = (100 / lenght) * poz; //кол-во процентов

            R = (int)((point & 0x00FF0000) >> 16);
            G = (int)((point & 0x0000FF00) >> 8);
            B = (int)((point & 0x000000FF) + N * 128 / 100);

            //контролируем переполнение переменных
            if (B < 0) B = 0;
            if (B > 255) B = 255;

            point = 0xFF000000 | ((UInt32)R << 16) | ((UInt32)G << 8) | ((UInt32)B);

            return point;
        }

    }
}
```

Файл BrightnessContrast.cs

```
using System;

namespace Program
{
    class BrightnessContrast
    {
        //якрость
        public static UInt32 Brightness (UInt32 point, int poz, int lenght)
        {
            int R;
            int G;
            int B;

            int N = (100 / lenght) * poz; //кол-во процентов

            R = (int)(((point & 0x00FF0000) >> 16) + N * 128 / 100);
            G = (int)(((point & 0x0000FF00) >> 8) + N * 128 / 100);
            B = (int)((point & 0x000000FF) + N * 128 / 100);

            //контролируем переполнение переменных
            if (R < 0) R = 0;
            if (R > 255) R = 255;
            if (G < 0) G = 0;
            if (G > 255) G = 255;
            if (B < 0) B = 0;
            if (B > 255) B = 255;

            point = 0xFF000000 | ((UInt32)R << 16) | ((UInt32)G << 8) | ((UInt32)B);

            return point;
        }
```

```
//контрастность
public static UInt32 Contrast(UInt32 point, int poz, int lenght)
{
    int R;
    int G;
    int B;

    int N = (100 / lenght) * poz; //кол-во процентов

    if (N >= 0)
    {
        if (N == 100) N = 99;
        R = (int)((((point & 0x00FF0000) >> 16) * 100 - 128 * N) / (100 - N));
        G = (int)((((point & 0x0000FF00) >> 8) * 100 - 128 * N) / (100 - N));
        B = (int)(((point & 0x000000FF) * 100 - 128 * N) / (100 - N));
    }
    else
    {
        R = (int)((((point & 0x00FF0000) >> 16) * (100 - (-N)) + 128 * (-N)) / 100);
        G = (int)((((point & 0x0000FF00) >> 8) * (100 - (-N)) + 128 * (-N)) / 100);
        B = (int)(((point & 0x000000FF) * (100 - (-N)) + 128 * (-N)) / 100);
    }

    //контролируем переполнение переменных
    if (R < 0) R = 0;
    if (R > 255) R = 255;
    if (G < 0) G = 0;
    if (G > 255) G = 255;
    if (B < 0) B = 0;
    if (B > 255) B = 255;

    point = 0xFF000000 | ((UInt32)R << 16) | ((UInt32)G << 8) | ((UInt32)B);
```

```
            return point;
        }

    }
}
```

Файл MatricaSvertki.cs

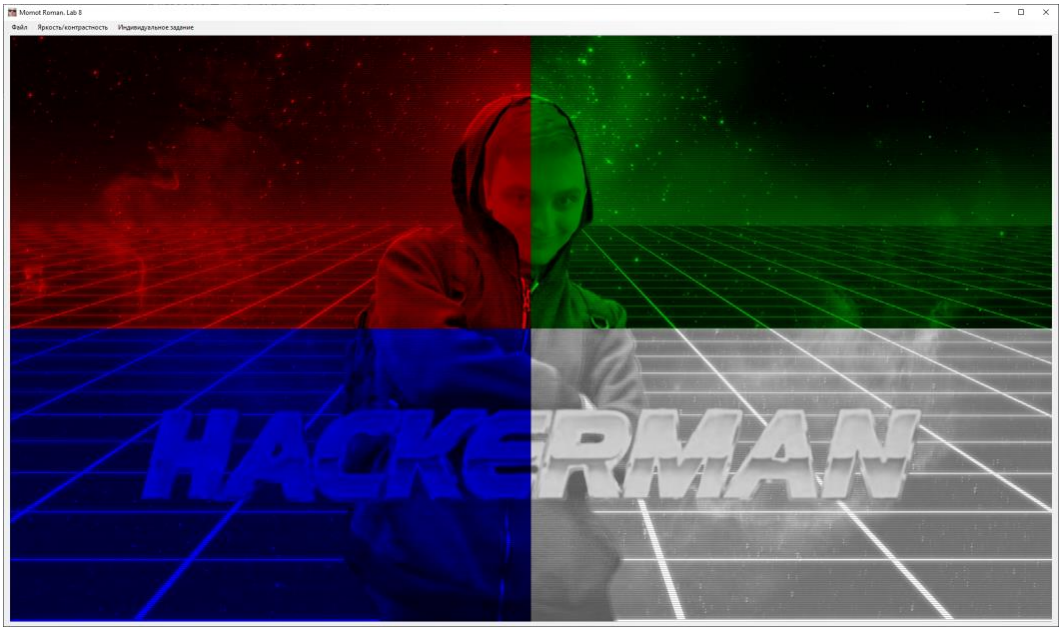
```
using System;

namespace Program
{
    struct RGB
    {
        public float R;
        public float G;
        public float B;
    }

    class Filter
    {
        public static UInt32[,] matrix_filtration(int W, int H, UInt32[,] pixel, int N,
double[,] matryx)
        {
            int i, j, k, m, gap = (int)(N / 2);
            int tmpH = H + 2 * gap, tmpW = W + 2 * gap;
            UInt32[,] tmppixel = new UInt32[tmpH, tmpW];
            UInt32[,] newpixel = new UInt32[H, W];
            //заполнение временного расширенного изображения
            //углы
            for (i = 0; i < gap; i++)
                for (j = 0; j < gap; j++)
                {
                    tmppixel[i, j] = pixel[0, 0];
```

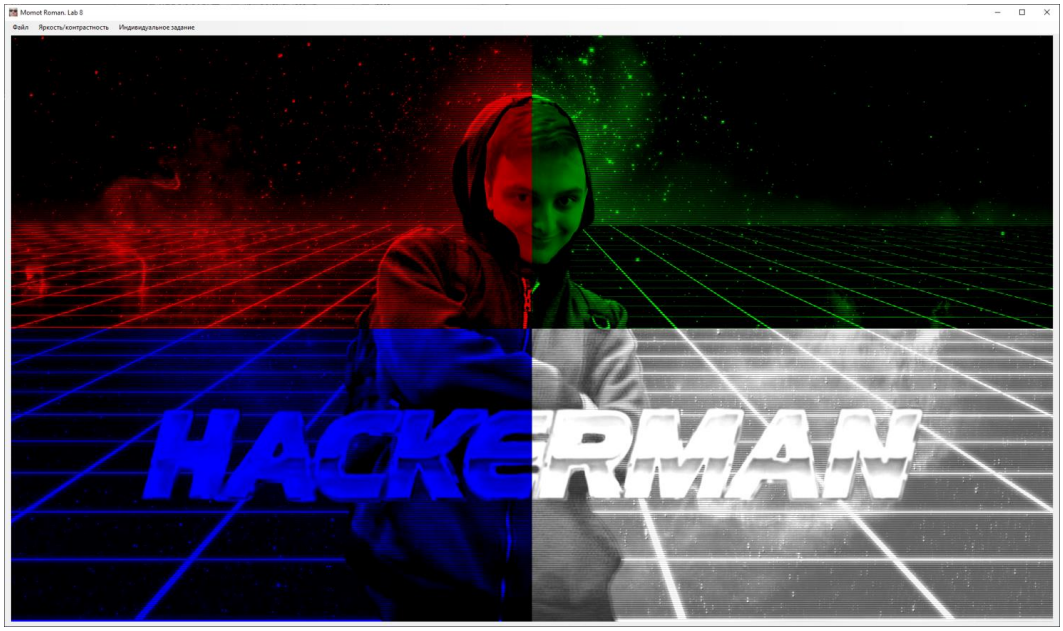
```
            tmppixel[i, tmpW - 1 - j] = pixel[0, W - 1];
            tmppixel[tmpH - 1 - i, j] = pixel[H - 1, 0];
            tmppixel[tmpH - 1 - i, tmpW - 1 - j] = pixel[H - 1, W - 1];
        }
    //крайние левая и правая стороны
    for (i = gap; i < tmpH - gap; i++)
        for (j = 0; j < gap; j++)
        {
            tmppixel[i, j] = pixel[i - gap, j];
            tmppixel[i, tmpW - 1 - j] = pixel[i - gap, W - 1 - j];
        }
    //крайние верхняя и нижняя стороны
    for (i = 0; i < gap; i++)
        for (j = gap; j < tmpW - gap; j++)
        {
            tmppixel[i, j] = pixel[i, j - gap];
            tmppixel[tmpH - 1 - i, j] = pixel[H - 1 - i, j - gap];
        }
    //центр
    for (i = 0; i < H; i++)
        for (j = 0; j < W; j++)
            tmppixel[i + gap, j + gap] = pixel[i, j];
    //применение ядра свертки
    RGB ColorOfPixel = new RGB();
    RGB ColorOfCell= new RGB();
    for (i = gap; i < tmpH - gap; i++)
        for (j = gap; j < tmpW - gap; j++)
        {
            ColorOfPixel.R = 0;
            ColorOfPixel.G = 0;
            ColorOfPixel.B = 0;
            for (k = 0; k < N; k++)
                for (m = 0; m < N; m++)
```

```
                {
                    ColorOfCell = calculationOfColor(tmppixel[i - gap + k, j - gap + m],
matryx[k, m]);
                    ColorOfPixel.R += ColorOfCell.R;
                    ColorOfPixel.G += ColorOfCell.G;
                    ColorOfPixel.B += ColorOfCell.B;
                }
            //контролируем переполнение переменных
            if (ColorOfPixel.R < 0) ColorOfPixel.R = 0;
            if (ColorOfPixel.R > 255) ColorOfPixel.R = 255;
            if (ColorOfPixel.G < 0) ColorOfPixel.G = 0;
            if (ColorOfPixel.G > 255) ColorOfPixel.G = 255;
            if (ColorOfPixel.B < 0) ColorOfPixel.B = 0;
            if (ColorOfPixel.B > 255) ColorOfPixel.B = 255;

            newpixel[i - gap, j - gap] = build(ColorOfPixel);
        }

    return newpixel;
}

//вычисление нового цвета
public static RGB calculationOfColor(UInt32 pixel, double coefficient)
{
    RGB Color = new RGB();
    Color.R = (float)(coefficient * ((pixel & 0x00FF0000) >> 16));
    Color.G = (float)(coefficient * ((pixel & 0x0000FF00) >> 8));
    Color.B = (float)(coefficient * (pixel & 0x000000FF));
    return Color;
}

//сборка каналов
public static UInt32 build(RGB ColorOfPixel)
```

```
            {
                UInt32 Color;
                Color = 0xFF000000 | ((UInt32)ColorOfPixel.R << 16) |
        ((UInt32)ColorOfPixel.G << 8) | ((UInt32)ColorOfPixel.B);
                return Color;
            }
        }
}
```

Результат роботи





Висновки

При виконанні даної лабораторної роботи було набуто навичок по роботі з растровими зображеннями.