МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІНСТИТУТ КОМП'ЮТЕРНИХ СИСТЕМ
КАФЕДРА КОМП'ЮТЕРНИХ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ І МЕРЕЖ

Протокол
Лабораторна робота №4
На тему "Реализация простых геометрических преобразований для растровых изображений"
По предмету: «Цифрова обробка сигналів та зображень»

Виконав:
студент групи АМ-182
Борщов М.І.
Перевірив:
Защолкін К. В.

Одеса 2022

ЗАВДАННЯ ДЛЯ РОБОТИ

Мета. Основы программной обработки растровых изображений.

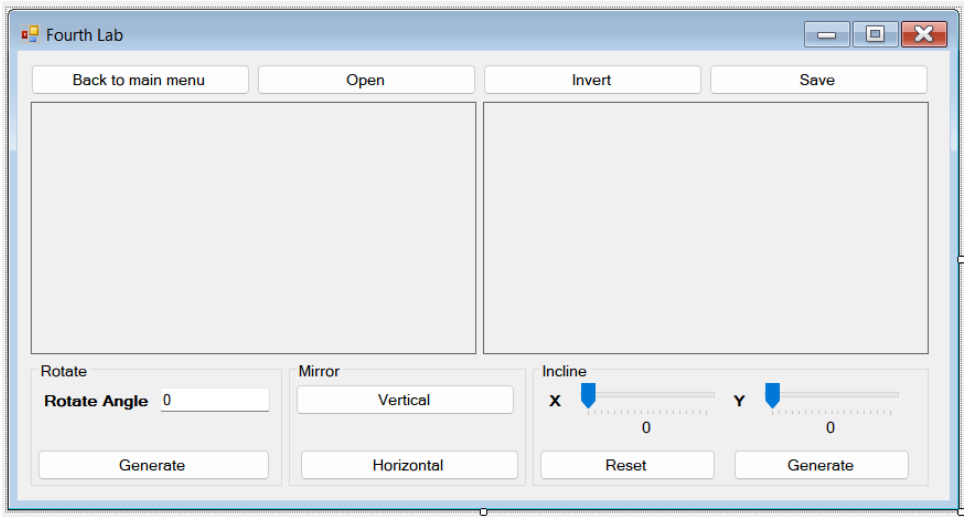
Завдання. В данной лабораторной работе необходимо дополнить приложение, разработанное в предыдущей работе, тремя функциональностями, обеспечивающими три рассмотренные геометрические преобразования растрового изображения.

Функциональность №1: по нажатию кнопки «Поворот», приложение выводит на экран, кроме основного изображения, его повернутый против часовой стрелки на выбранный угол вариант. Пользователь приложения должен иметь возможность задать угол поворота в диапазоне от 0□ до 360□. Для задания угла рекомендуется использовать компонент TextBox. Необходимо предусмотреть контроль ввода символов в этот компонент, а именно: – количество символов не должно превышать трех; – компонент не должен давать пользователю возможность вводить символы отличные от цифр 0 – 9; – при попытке ввести значение большее 360, содержимое компонента TextBox должно устанавливаться в значение 360.

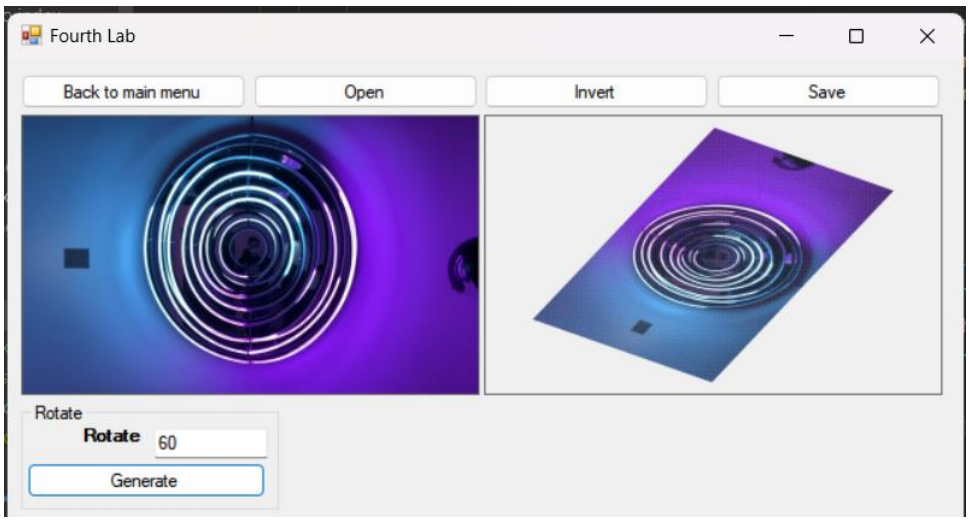
Функциональность №2: по нажатию кнопки «Отражение» приложение выводит на экран отраженное слева направо или сверху вниз изображение. Пользователь приложения должен иметь возможность при помощи компонента RadioButton выбрать вариант отражения (слева направо или сверху вниз).

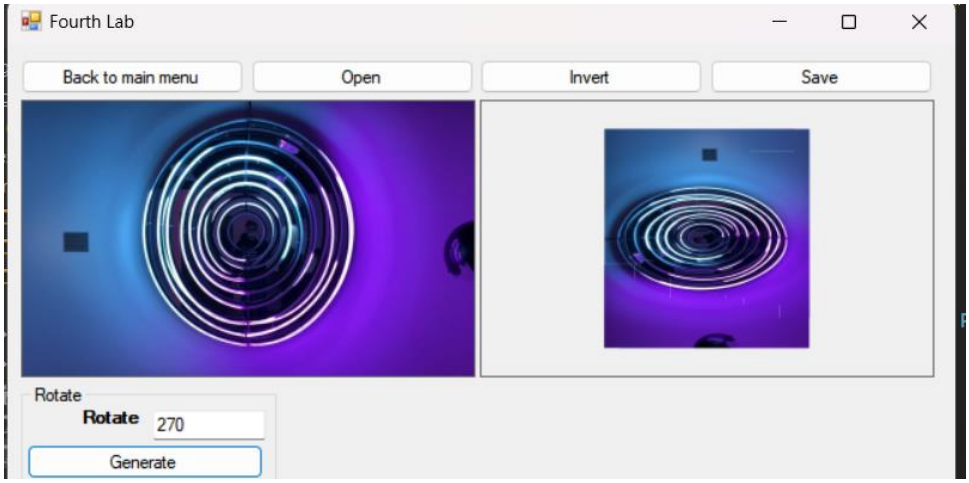
Функциональность №3: по нажатию кнопки «Наклон» приложение выводит на экран наклоненное изображение. Пользователь приложения должен иметь возможность при помощи компонентов TrackBar выбрать величину наклона по горизонтали и по вертикали.

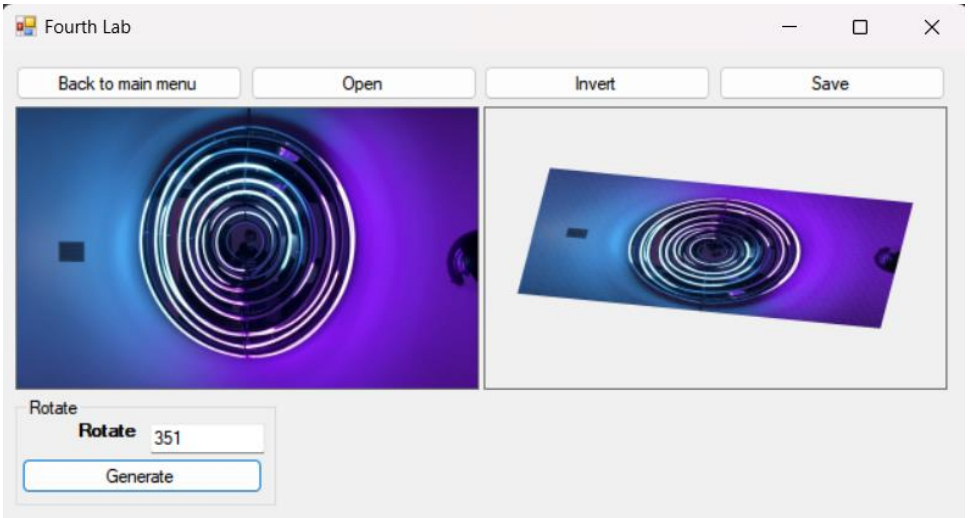
Хід роботи



Дизайн програми







Результат повороту на заданий кут проти годинникової стрілки

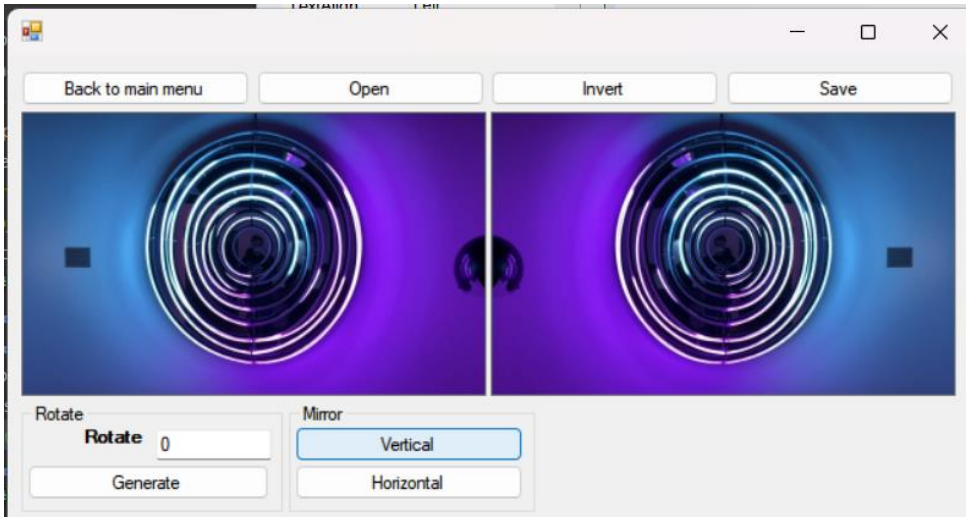
Зображення трохи деформується, бо поле з результуючою картинкою має відмінне від зображення співвідношення сторін

```
private void rotateTextBox_TextChanged(object sender, EventArgs e)
{
    if (!rotateTextBox.Text.All(char.IsDigit)) return;
    try
    {
        var rotate :int = int.Parse(rotateTextBox.Text);
        if (rotate > 360)
        {
            rotateTextBox.Text = "360";
            rotate = 360;
        }

        else if (rotate < 0)
        {
            rotateTextBox.Text = "0";
            rotate = 0;
        }

        _rotateValue = rotate * Math.PI / 180;
    }
    catch (FormatException)
    {
        _rotateValue = 0;
        rotateTextBox.ResetText();
    }
}
```

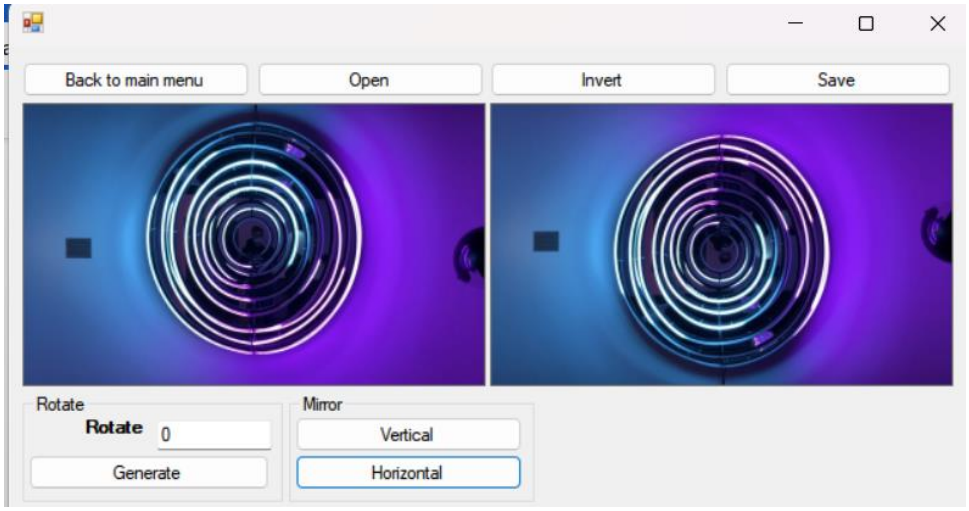
Код функції задання куту повороту та переведення його у радіани

```
if (_originalImage == null) return;

// generate possible largest image when rotate
var hypotenuse = (int)Math.Sqrt(_originalImage.Width * _originalImage.Width +
                                _originalImage.Height * _originalImage.Width);
_generatedImage = new Bitmap( width: hypotenuse,  height: hypotenuse);

var originalCentralX :int  = _originalImage.Width / 2;
var originalCentralY :int  = _originalImage.Height / 2;
var generatedCentralX :int  = _generatedImage.Width / 2;
var generatedCentralY :int  = _generatedImage.Height / 2;
var moveX :int  = generatedCentralX - originalCentralX;
var moveY :int  = generatedCentralY - originalCentralY;

for (var i = 0; i < _originalImage.Width; i++)
for (var j = 0; j < _originalImage.Height; j++)
{
    var x :double  = (i - originalCentralX) * Math.Cos(_rotateValue) -
        (j - originalCentralY) * Math.Sin(_rotateValue) + originalCentralX;
    var y :double  = -(i - originalCentralX) * Math.Sin(_rotateValue) -
        (j - originalCentralY) * Math.Cos(_rotateValue) + originalCentralY;

    // add moveX and moveY for moving rotated original image to the center of generated
    _generatedImage.SetPixel( x: (int)x + moveX,  y: (int)y + moveY,  color: _originalImage.GetPixel( x: i,  y: j));
}

generatedPicture.Image = _generatedImage;
```

Код повороту зображення



Вертикальне відображення

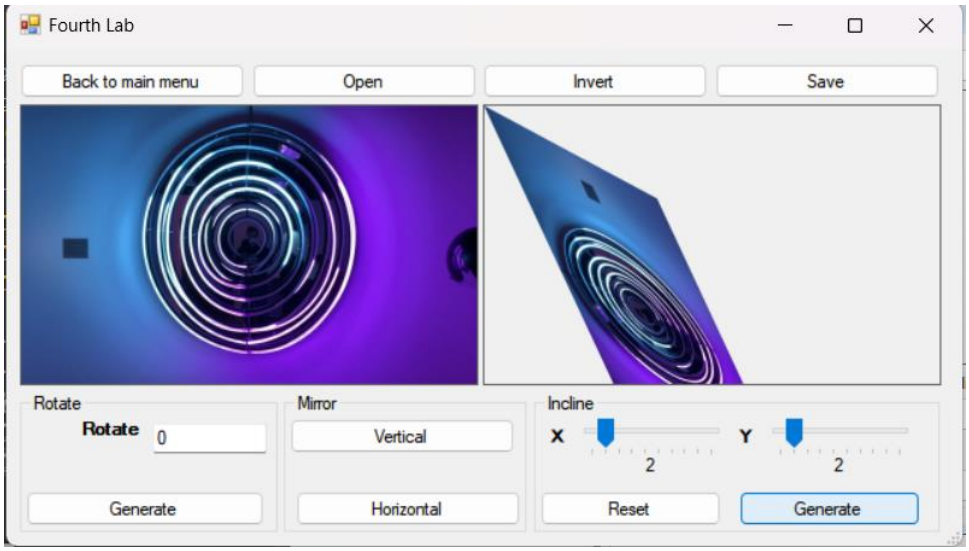


Горизонтальне відображення

```
1 usage
private void mirrorVerticalButton_Click(object sender, EventArgs e)
{
    if (_originalImage == null) return;

    _generatedImage = new Bitmap(_originalImage);

    for (var i = 0; i < _generatedImage.Width; i++)
    for (var j = 0; j < _generatedImage.Height; j++)
    {
        var x:int = _generatedImage.Width - i;
        try
        {
            _generatedImage.SetPixel(x, y: j, color: _originalImage.GetPixel(x: i, y: j));
        }
        catch (ArgumentOutOfRangeException)
        {
        }
    }

    generatedPicture.Image = _generatedImage;
}
```

```
private void mirrorHorizontalButton_Click(object sender, EventArgs e)
{
    if (_originalImage == null) return;

    _generatedImage = new Bitmap(_originalImage);

    for (var i = 0; i < _generatedImage.Width; i++)
    for (var j = 0; j < _generatedImage.Height; j++)
    {
        var y:int = _generatedImage.Height - j;
        try
        {
            _generatedImage.SetPixel(x: i, y, color: _originalImage.GetPixel(x: i, y: j));
        }
        catch (ArgumentOutOfRangeException)
        {
        }
    }

    generatedPicture.Image = _generatedImage;
}
```

Код відображення



Результат нахилу

```
if (_originalImage == null) return;
// handle divide by zero
if (inclineXArgTrackBar.Value == 1 && inclineYArgTrackBar.Value == 1)
{
    generatedPicture.Image = _originalImage;
    return;
}
_generatedImage = new Bitmap(_originalImage.Width, _originalImage.Height);
for (var i = 0; i < _originalImage.Width; i++)
for (var j = 0; j < _originalImage.Height; j++)
{
    // i = x + ay  ;  j = bx + y
    // -> x = ay - i; y = bx - j
    // --> y = b(ay - i) - j  -->  y - bay = -bi - j -->  y(ba - 1) = bi + j
    // --> y = (bi + j) / (ba - 1)
    //     x = ay - i
    var y :int = (inclineXArgTrackBar.Value * i + j) /
            (inclineXArgTrackBar.Value * inclineYArgTrackBar.Value - 1);
    var x :int = inclineYArgTrackBar.Value * y - i;
    try
    {
        _generatedImage.SetPixel(x, y, color: _originalImage.GetPixel( x: i, y: j));
    }
    catch (ArgumentOutOfRangeException)
    {
    }
}

generatedPicture.Image = _generatedImage;
```

Код нахилу

Також були оброблені ситуації, коли зображення ще не було відкрите, але користувач намагається викликати інші функції

Код програми: https://github.com/zeinlol/image-converter