Photoshop App Simulator with C++

I. Algorithms of filters
II. Function diagram

By : Tarek Muhammed Abdel-Hamid Muhammed

GREY SCALE FUNCTIONS EXPLAINING:

- void loadImage (); //load image
- void loadImage2 (); //load second image to use in merge filter
- void saveImage (); //save image after editing
- void BlackandWhite(); //black and white filter
- void Invert(); //invert image
- void Merge2photos(); //merge two images
- void Flip(); //flip an image horizontally and vertically
- void Rotate(); //rotate an image
- void Brightness(); // dark by 50% or light by 50%
- void DetectEdges(); // detect edges of an image
- void Enlarge(); //enlarge a quarter of an image
- void Blur(); // blur image
- void Shuffle(); //shuffle image
- void Mirroring(); //mirror image
- void Crop(); //crop image
- void Shrink()/shrink an image
- void Skew(); //Skew an Image horizontally and vertically
- void ChooseEffect();//Choose the effect you want to apply
- void KeepEditing();//Choose Multible Effects

I. Controlling Functions:

void loadImage();
1.Array of 100 characters for the image file name
2.Input image file name
3.Takes the image and convert it to .bmp
void loadImage2();
loadimage() same algorithm, we will use it in merge function only.

void saveImage();
1.take a name of image from user
2.strcat is a standard library function used to concatenate two strings and appending the ".bmp" string to the end.

void ChooseEffect();
A function based on switch cases to make the user choose the effect he want to apply in easy way .

the first the user input the number of the choice he wants even he want to leave the program

void keepEditing();
A function to make the user apply more than effect in the same time by sending the function of choose effect again if he want it or save the photo and end the program .

II. Filters Functions:

void BlackandWhite();
1.two four loops to access each index of the 2d array
2.if the grey level in a pixel more than 127 → will assign it to 255 (white)
3.if the grey level in a pixel less than 127 → will assign it to 0 (black)

void Invert();
1.two four loops to access each element of the 2d array
2.the new value of the pixel will be (255-pixel value)

void Merge2Photos();
1.will take the second image from the user using loadimage2().
2.(pixel of image1 + pixel of image2)/2 to merge both of them together.

void Flip();
1.if we want to flip it vertically :
will change the row index from I to size-I with the same column.
2.if we want to flip it horizontally :
will flip vertically then rotate 180 degrees.

void Rotate();
will take the degree we want to rotate to from the user
1.if it's 90: the new indices will change from [i][j] to [j][size-i-1]
2.if it's 180 : change [i] to [size – i– 1] and [j] to [size – j – 1]
3.if it's 270 : will rotate 90 three times

void Brightness();
1.will take the photo from the user
2.if we want to darken it the user will enter 1:
3.then we will devide by two
4.if we want to lighten it the user will enter 2:
5.then we will add 255 then devide by two

void DetectEdges();
1.we will convert it to black and white photo
2.then we will compore the pixel with the pixel that next in the same column
3.then we will compare the pixel with the pixel that next in the same row
4.if one of them satisfy the condition the pixel will be 0

void Enlarge();
1.the user will enter the the place of the quarter he wants
2.then we will make a photo that contains that quarter
3.then we will enlarge it

void Blur();
1.Variable for storing blurry image.
2.Two for loops so that you can access each index.
3.loop for blurring the image multiple times 10 in this case.
4.variable for sum and count of color channels.
5.iterate through a 3*3 neighborhood around the current pixel.
6.calculate the average values for each color channel.
7.store the blurry image in the original image.

void Shuffle();

1. Input Sequence of Parts
2. Check Validity of Input , Copy Image Parts to Temporary Arrays
3. Apply Selected Shuffling Sequence

- Step for Part 1
- Step for Part 2
- Step for Part 3
- Step for Part 4

4. Update Original Image with Shuffled Parts

void Mirroring();

1. Input Mirroring Type
2. Loop through image rows and columns
3. Apply type of mirroring

void Shrink();

1. Store the original image in a new image.
2. Make the original image white.
3. If shrink ½ : loop for original image /2 and sore the new image in it (index *2).
4. Else 4.if shrink 1/3 : loop for original image /3 and sore the new image in it)index*3).
5. Else 5.if shrink 1/4: loop for original image /4 and sore the new image in it)index*4).

void Crop();

1. Input Parameters

x: X-coordinate of top-left corner

y: Y-coordinate of top-left corner

l: Length of the crop

w: Width of the crop

2. Check Boundary Conditions

If crop exceeds image boundaries, display a warning message.

3. Initialize New Image Array

Create a new array to hold the cropped image.

4. Iterate Through Image Pixels

Nested loops to traverse through each pixel in the original image.

5. Crop Region Condition

Check if the pixel is within the specified crop region.

6. Copy Pixel Values

If within crop region, copy the pixel value to the new image.

7. Set Background Value

If outside the crop region, set the pixel value to 255 (background).

8. Copy New Image to Original

Update the original image with the cropped version.

void Skew();

1.Temporary image for skewed image

2.Calculate side to leave unaffected by skewscaling factor.

3.Step size

4.Variables for tracking position and pixels

5.Iterate through columns and rows to perform skewing.

6.Iterate over pixels to calculate average.

7.Calculate and store the average.

8.Update pixels taken.

9.Copy the skewed image back to the original image.

III. Functions Diagram: