Histogramequalization.cpp

```cpp
#include "stdafx.h"
#include <opencv2/opencv.hpp>
#include <iostream>
#include <list>
#include "Intensity.h"
#include "cmath"

using namespace std;
using namespace cv;

//Creates a histogram image based on image input "newHistogram(Mat image)"
Mat newHistogram(Mat& original) {
    //Creates a new placeholder for the histogram depection of the equalized image
    Intensity* histogramArray[255];

    //The following code is the same procedure as the previous histogram example.
    for (int i = 0; i < 256; i++)
        histogramArray[i] = new Intensity();


    //Goes through placeholder and sets the amount of each intensity to the respective value
    //of the original image
    for (int i = 0; i < original.cols; i++)
        for (int j = 0; j < original.rows; j++)
            histogramArray[original.at<uchar>(j, i)]->set_amount();

    //Creating the visualization of the histogram from the loaded image
    int hist_w = 512; int hist_h = 400;
    //bin_w is the amount of bins in the histogram. As it is intended to depict the various intensity's with their
    //respective pixel amount's, the length of the depicted histogram is devided by this value.
    int bin_w = cvRound((double)hist_w / 256);

    //Initialising the Mat object named returnedHistImage (The basic Image container). This is an object that will store the information
    //of the visual representation of the histogran for the loaded image.
    Mat returnedHistImage(hist_h, hist_w, CV_8UC1, Scalar(255, 255, 255));
    //CV: CV_[The number of bits per item][Signed or Unsigned][Type Prefix]C[The channel number]
    //So a image container with 8 bits per pixel, unsigned, and 1 channel is created

    //Goes through placeholder. Finds the intensity with the highest amount of pixels with the respective intensity.
    int hist_max = histogramArray[0]->get_amount();
    for (int i = 1; i < 256; i++)
        if (hist_max < histogramArray[i]->get_amount())
            hist_max = histogramArray[i]->get_amount();

    //Goes through placeholder. With the knowledge of the highest amount, a normalisation of the histogram array
    //is done to vertically fit within the visualisation of the histogram. This new intensity value
    //is stored in the intensity object's histAmount variable.
    for (int i = 0; i < 256; i++) {
        int temp = ((double)histogramArray[i]->get_amount() / hist_max)*returnedHistImage.rows;
        histogramArray[i]->set_histAmount(temp);
    }

    //Creates each bin in the visualization with the use of points provided by openCV
    for (int i = 0; i < 256; i++)
        line(returnedHistImage, Point(bin_w*(i), hist_h),
            Point(bin_w*(i), hist_h - histogramArray[i]->get_histAmount()),
            Scalar(0, 0, 0), 1, 8, 0);

    //Returns the created histogram from the given image-input
    return returnedHistImage.clone();
}

int main() {
    //Loads input image
    Mat src_img = imread("grayscale.jpg", CV_LOAD_IMAGE_GRAYSCALE);
    //Creates the window to store the image
    char* source_image = "Source image";
    namedWindow(source_image, CV_WINDOW_AUTOSIZE);
    //Shows the image in the specified window
    imshow(source_image, src_img);

    //Creates a placeholder for the various bins[0, 255] in a histogram made up from the previous loaded image
    //This placeholder will work as a lookup table to find an equalized value for an intensity in an image
    Intensity* src_lookup[255];

    //Goes through placeholder and instantiates a Intensity object which holds the variables:
    //(int) amount, (float) probability, (float) culmProbability, and (int) final_culmProbability.
    for (int i = 0; i < 256; i++)
        src_lookup[i] = new Intensity();

    //Goes through placeholder. Depending on the value (intensity) of the pixel at: img.at<uchar>(j, i)
    //an array element is selected and the function set_amount() is called upon this element.
    //Meaning if the pixel's intensity is 5, element src_lookup[5] will be chosen.
    //set_amount() increases the count of the various intensities.
    for (int i = 0; i < src_img.cols; i++)
        for (int j = 0; j < src_img.rows; j++)
            src_lookup[src_img.at<uchar>(j, i)]->set_amount();
    //The lookup table now knows the amount of each intensity in the loaded image

    //Goes through placeholder. For each intensity element, the respective probability is calcuted.
    //This is calculated by taking a respective intensity and dividing it by the amount of
    //pixels in the image. Size of image is sent through the function call: set_probability(image_size)
    for (int i = 0; i < 256; i++)
        src_lookup[i]->set_probability(src_img.rows*src_img.cols);
    //The lookup table now knows the probability of each intensity-value in the loaded image

    //Goes through placeholder. A culminated probability is calculated. To start the calculation a temp variable is
    //passed through the method call set_culmProbability(previous_culminated_probability).
    //To begin the calculation this variable is initiliased as 0.
    float variableholder = 0;
    for (int i = 0; i < 256; i++)
        variableholder = src_lookup[i]->set_culmProbability(variableholder);
    //The histogram now knows the culminated probability of each intensity-value in the loaded image

    //Goes through spaceholder. Based on each intensity value's culminated probability, a look up table is created
    //by multiplying each culminated probability with the intented maximum value. In this case 255.
    //This means that based on the intensity's probability a new intensity value is given.
    //E.g. if a pixels intensity value is 50, the lookup table (based on probability of that given intensity)
    //will return a new value for this intensity, e.g. 22
    for (int i = 0; i < 256; i++)
        src_lookup[i]->set_final_culmProbability();

    //Creates a histogram-visulization of the source image
    Mat histImage = newHistogram(src_img);
    //Creates the window to store and show the histogram visualization
    char* histogram_window = "Source image - histogram";
    namedWindow(histogram_window, CV_WINDOW_AUTOSIZE);
    //Shows histogram image in the histogram window
    imshow(histogram_window, histImage);

    //Sets up a placeholder for the equalized image with the same size and settings as the original image
    Mat equalizationImage(src_img.rows, src_img.cols, CV_8UC1, Scalar(255, 255, 255));

    //Goes through the original image and uses the look-up table created to chose the new equalized intensity for
    //the respective pixel
    for (int i = 0; i < src_img.rows; i++)
        for (int j = 0; j < src_img.cols; j++) {
            int prev_intensity = (int)src_img.at<uchar>(i, j);
            int new_intensity = src_lookup[prev_intensity]->get_final_culmProbability();
            equalizationImage.at<uchar>(i, j) = new_intensity;
        }

    //Creates the window to store the equalized image
    char* equalization_window = "Equalization image";
    namedWindow(equalization_window, CV_WINDOW_AUTOSIZE);
    //Shows the equalized image
    imshow(equalization_window, equalizationImage);

    //Creates a histogram for the equliazed image and stores it in a Image placeholder
    Mat eqHist = newHistogram(equalizationImage);
    //Creates the window to store the histogram of the equalized image
    char* eq_histogram_window = "Equalization image - histogram";
    namedWindow(eq_histogram_window, CV_WINDOW_AUTOSIZE);
    //Shows the histogram of the equalized image
    imshow(eq_histogram_window, eqHist);

    waitKey(0);
    return 1;
}
```

Intensity.h

```cpp
#pragma once
#include "cmath"
//header for the class Intensity
class Intensity
{
    int amount = 0;
    int final_culmProbability, hist_amount;
    float probability, culmProbability;
public:
    void set_amount();
    int get_amount();
    void set_histAmount(int n_amount);
    int get_histAmount();
    void set_probability(int);
    float get_probability();
    float set_culmProbability(float);
    float get_culmProbability();
    void set_final_culmProbability();
    int get_final_culmProbability();
};
```

Intensity.cpp

```cpp
#include "stdafx.h"
//include header of this class
#include "Intensity.h"

//function that increases objects amount count
void Intensity::set_amount() {
    amount++;
}

//function that returns the count of the intensity object's amount variable
int Intensity::get_amount() {
    return amount;
}

//function that changes the Intensity object's variable histAmount with a specified int variable
void Intensity::set_histAmount(int n_amount) {
    hist_amount = n_amount;
}

//function that returns the count of the intensity object's histAmount variable
int Intensity::get_histAmount() {
    return hist_amount;
}

//function that calculates and sets the probability of an Intensity's value
//by dividing the count of the intensity-value by the given image's pixel amount
//which is passed through the function call
void Intensity::set_probability(int pixelAmount) {
    probability = (float)amount / pixelAmount;
}

//function that returns the probability of the intensity-value
float Intensity::get_probability() {
    return probability;
}

//function that calculates and sets the culminated probability
float Intensity::set_culmProbability(float prevCulmProbability) {
    culmProbability = prevCulmProbability + probability;
    set_final_culmProbability();
    return culmProbability;
}

//function that returns the culminated probability
float Intensity::get_culmProbability() {
    return culmProbability;
}

//function that calculates and sets the final culminated probability or the new equalized value for
//the intensity-value
void Intensity::set_final_culmProbability() {
    final_culmProbability = floor(culmProbability * 255);
}

//function that returns the equalized intensity-value
int Intensity::get_final_culmProbability() {
    return final_culmProbability;
}
```