**Question 1:** Read and print the pixel values of an image

**Code:**

#include <opencv2/opencv.hpp>

#include <iostream>

#include <string>

int main()

{

std::string filename;

std::cin >> filename;

cv::Mat img = cv::imread(filename);

int rows = img.rows, cols = img.cols \* img.channels();

std::cout << img.rows << '\*' << img.cols << " Channels: " << img.channels() << '\n';

const uchar\* ptr;

if(img.isContinuous())

{

cols \*= rows; rows = 1;

}

for(int i = 0; i < rows; i++)

{

ptr = img.ptr<uchar>(i);

for(int j = 0; j < cols; j++)

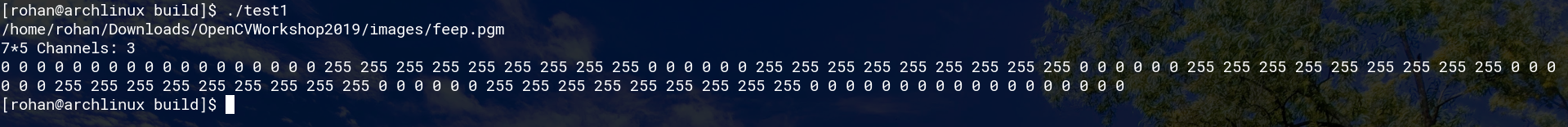
std::cout << (int)ptr[j] << ' ';

std::cout << '\n';

}

}

**Output:**



**Question 2: Print histogram of pixel values of a grey image**

**Code:**

#include <opencv2/opencv.hpp>

#include <iostream>

#include <vector>

#include <array>

int main()

{

cv::Mat src = cv::imread("/home/rohan/Downloads/OpenCVWorkshop2019/images/sharpening.png");

std::vector<cv::Mat> bgr\_planes;

cv::split(src, bgr\_planes);

int histSize = 256;

std::array<float, 2> range = {0, 256};

const float\* histRange = {range.data()};

cv::Mat b\_hist, g\_hist, r\_hist;

cv::calcHist(&bgr\_planes[0], 1, 0, cv::Mat(), b\_hist, 1, &histSize, &histRange);

cv::calcHist(&bgr\_planes[1], 1, 0, cv::Mat(), g\_hist, 1, &histSize, &histRange);

cv::calcHist(&bgr\_planes[2], 1, 0, cv::Mat(), r\_hist, 1, &histSize, &histRange);

int hist\_w = 512, hist\_h = 400;

int bin\_w = cvRound((double)hist\_w / histSize);

cv::Mat histImage(hist\_h, hist\_w, CV\_8UC3, cv::Scalar(0, 0, 0));

cv::normalize(b\_hist, b\_hist, 0, histImage.rows, cv::NORM\_MINMAX, -1, cv::Mat());

cv::normalize(g\_hist, g\_hist, 0, histImage.rows, cv::NORM\_MINMAX, -1, cv::Mat());

cv::normalize(r\_hist, r\_hist, 0, histImage.rows, cv::NORM\_MINMAX, -1, cv::Mat());

for(int i = 1; i < histSize; i++)

{

cv::line(histImage, cv::Point(bin\_w \* (i - 1), hist\_h - cvRound(b\_hist.at<float>(i - 1))),

cv::Point(bin\_w \* (i), hist\_h - cvRound(b\_hist.at<float>(i))), cv::Scalar(255, 0, 0));

cv::line(histImage, cv::Point(bin\_w \* (i - 1), hist\_h - cvRound(g\_hist.at<float>(i - 1))),

cv::Point(bin\_w \* (i), hist\_h - cvRound(g\_hist.at<float>(i))), cv::Scalar(0, 255, 0));

cv::line(histImage, cv::Point(bin\_w \* (i - 1), hist\_h - cvRound(r\_hist.at<float>(i - 1))),

cv::Point(bin\_w \* (i), hist\_h - cvRound(r\_hist.at<float>(i))), cv::Scalar(0, 0, 255));

}

cv::imshow("Source Image", src);

cv::imshow("calcHist Demo", histImage);

cv::waitKey();

}

**Output:**



**Question 3:** Display inverted gray image

**Code:**

#include <iostream>

#include <string>

#include <opencv2/opencv.hpp>

int main()

{

std::string filename; std::cin >> filename;

cv::Mat img = cv::imread(filename), img\_gray;

cv::imshow("original", img);

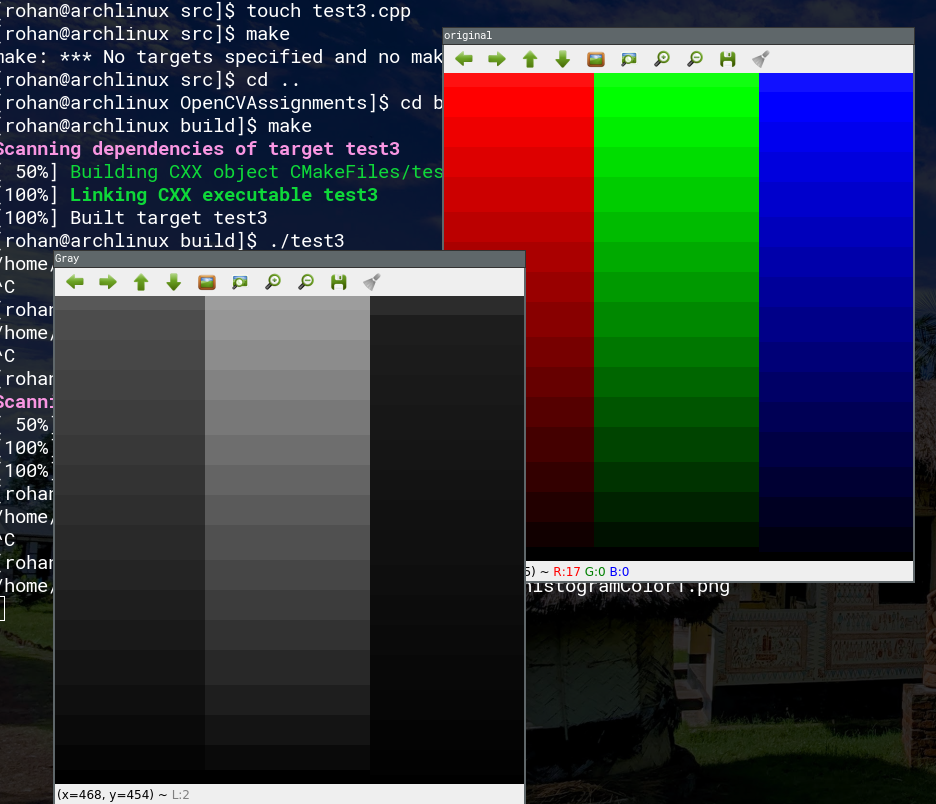
cv::cvtColor(img, img\_gray, cv::COLOR\_BGR2GRAY);

cv::imshow("Gray", img\_gray);

cv::waitKey();

}

**Output:**



**Question 4:** Enhance overall brightness of a grey image by a given value.

**Code:**

#include <opencv2/opencv.hpp>

#include <iostream>

#include <string>

int main()

{

std::string filename; std::cin >> filename;

cv::Mat img = cv::imread(filename);

float beta; std::cin >> beta;

cv::imshow("Original", img);

cv::Mat lookupTable(1, 256, CV\_8U);

uchar\* p = lookupTable.ptr();

for(int i = 0; i < 256; i++)

p[i] = cv::saturate\_cast<uchar>(i + beta);

cv::LUT(img, lookupTable, img);

cv::imshow("Altered Image", img);

cv::waitKey();

}

**Output:**



**Question 5:** Enhance contrast of a grey image.

**Code:**

#include <opencv2/opencv.hpp>

#include <iostream>

#include <string>

int main()

{

std::string filename; std::cin >> filename;

cv::Mat img = cv::imread(filename);

float alpha; std::cin >> alpha;

cv::imshow("Original", img);

cv::Mat lookupTable(1, 256, CV\_8U);

uchar\* p = lookupTable.ptr();

for(int i = 0; i < 256; i++)

p[i] = cv::saturate\_cast<uchar>(i \* alpha);

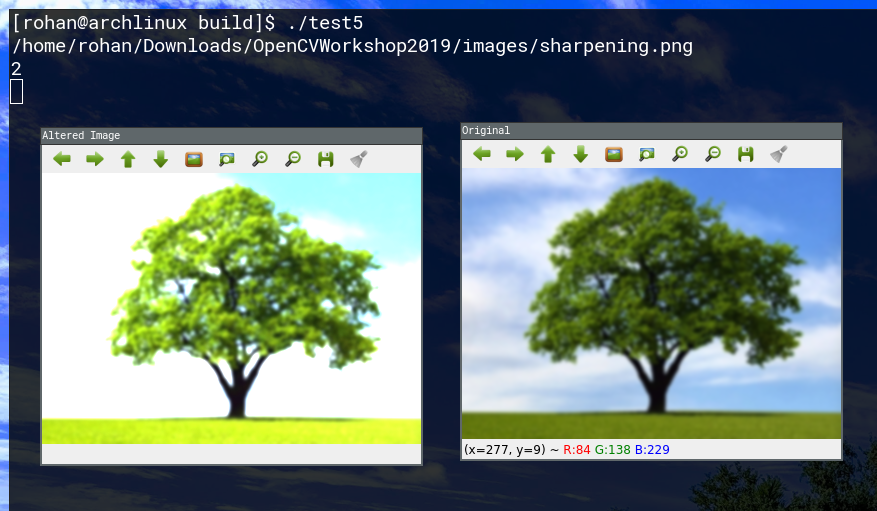
cv::LUT(img, lookupTable, img);

cv::imshow("Altered Image", img);

cv::waitKey();

}

**Output:**



**Question 6:** Perform histogram equalization to enhance quality of a grey image.

**Code:**

#include <opencv2/opencv.hpp>

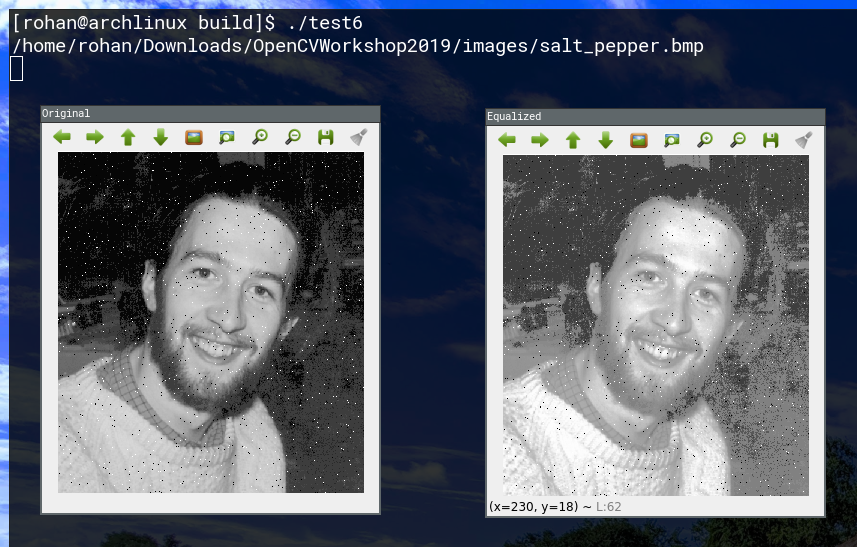
#include <iostream>

#include <string>

int main()

{

std::string filename; std::cin >> filename;

 cv::Mat img = cv::imread(filename);

cv::cvtColor(img, img, cv::COLOR\_BGR2GRAY);

cv::imshow("Original", img);

cv::Mat m\_img;

cv::equalizeHist(img, m\_img);

cv::imshow("Equalized", m\_img);

cv::waitKey();

}

**Output:**

**Question 7:** Generate a binary thresholded image from a grey image.

**Code:**

#include <opencv2/opencv.hpp>

#include <iostream>

cv::Mat img, img1;

int threshVal = 127;

int maxThreshVal = 255;

void callback(int, void\*)

{

cv::threshold(img, img1, threshVal, maxThreshVal, cv::ThresholdTypes::THRESH\_BINARY);

cv::imshow("Thresholded", img1);

}

int main()

{

std::string filename; std::cin >> filename;

img = cv::imread(filename);

cv::cvtColor(img, img, cv::COLOR\_BGR2GRAY);

cv::imshow("Original", img);

cv::namedWindow("Thresholded");

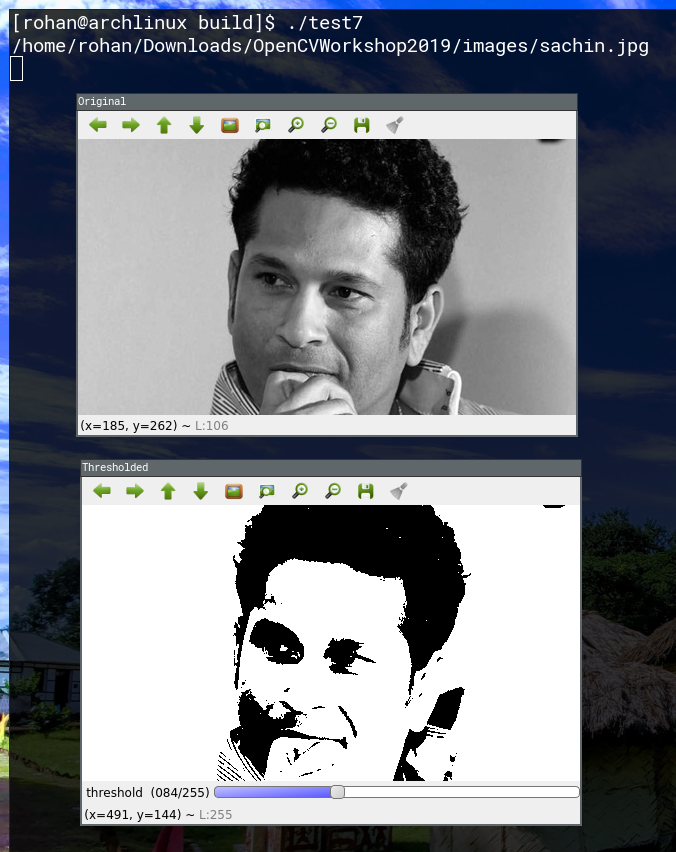
cv::createTrackbar("threshold", "Thresholded", &threshVal, maxThreshVal, callback);

callback(threshVal, 0);

cv::waitKey();

}

**Output:**



**Question 8:** Zoom in and zoom out a grey image by a given scale factor.

**Code:**

#include <opencv2/opencv.hpp>

#include <iostream>

cv::Mat img, img1;

int val = 5;

int maxVal = 10;

void callback(int, void\*)

{

double factor = (double)val / (maxVal / 2);

cv::resize(img, img1, cv::Size(), factor, factor);

cv::imshow("zoomed", img1);

}

int main()

{

std::string filename; std::cin >> filename;

img = cv::imread(filename);

cv::imshow("original", img);

cv::namedWindow("zoomed");

cv::createTrackbar("scale", "zoomed", &val, maxVal, callback);

callback(val, 0);

cv::waitKey();

}

**Output:**

