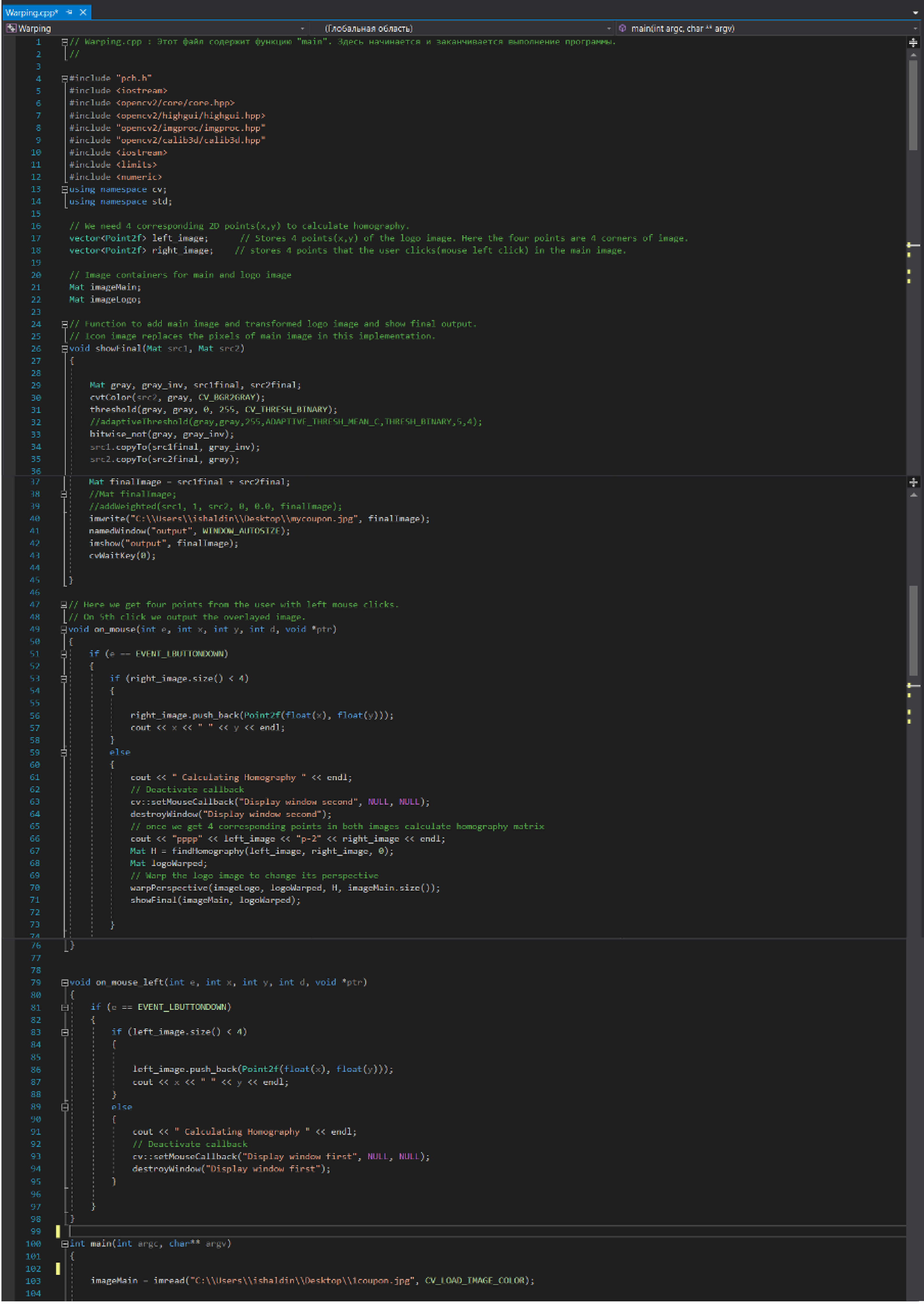
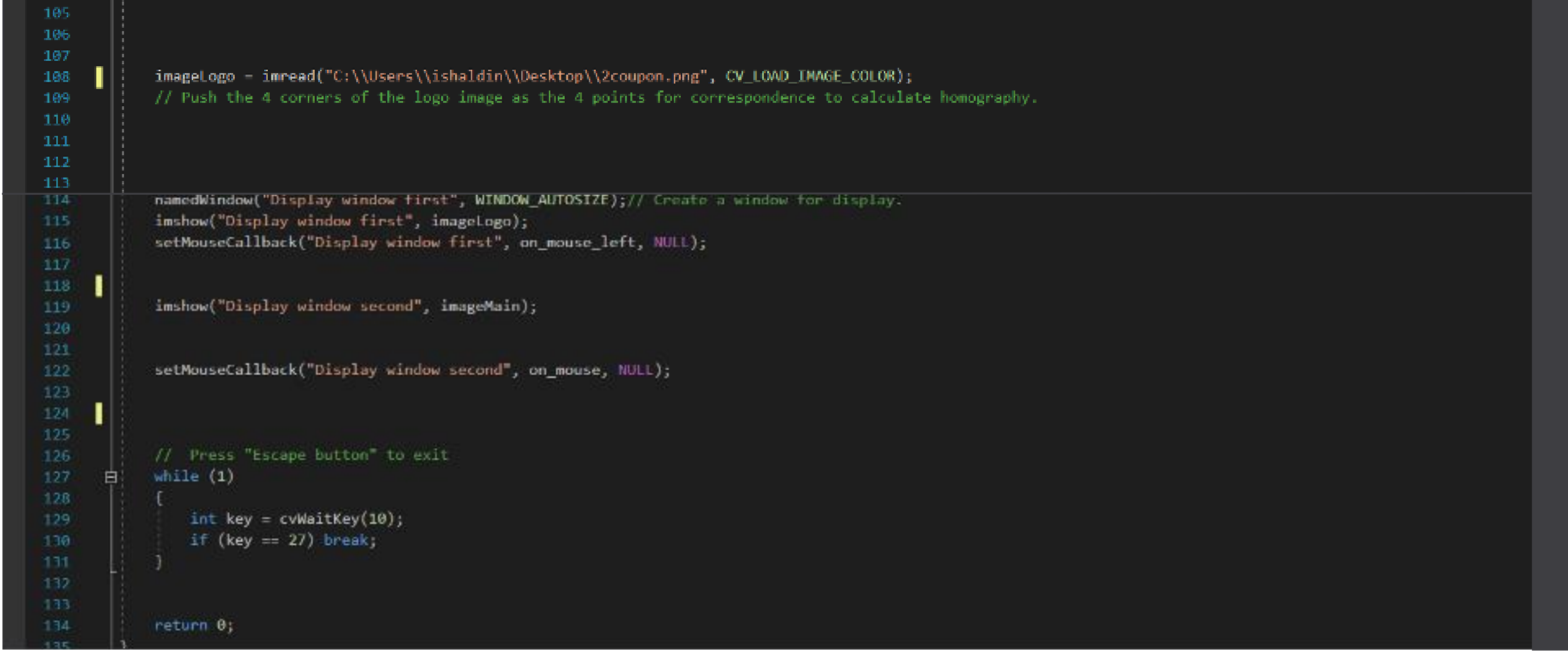


```cpp
// Warping.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.
//

#include "pch.h"
#include <iostream>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/calib3d/calib3d.hpp"
#include <iostream>
#include <limits>
#include <numeric>
using namespace cv;
using namespace std;

// We need 4 corresponding 2D points(x,y) to calculate homography.
vector<Point2f> left_image;      // Stores 4 points(x,y) of the logo image. Here the four points are 4 corners of image.
vector<Point2f> right_image;    // stores 4 points that the user clicks(mouse left click) in the main image.

// Image containers for main and logo image
Mat imageMain;
Mat imageLogo;

// Function to add main image and transformed logo image and show final output.
// Icon image replaces the pixels of main image in this implementation.
void showFinal(Mat src1, Mat src2)
{


    Mat gray, gray_inv, src1final, src2final;
    cvtColor(src2, gray, CV_BGR2GRAY);
    threshold(gray, gray, 0, 255, CV_THRESH_BINARY);
    //adaptiveThreshold(gray,gray,255,ADAPTIVE_THRESH_MEAN_C,THRESH_BINARY,5,4);
    bitwise_not(gray, gray_inv);
    src1.copyTo(src1final, gray_inv);
    src2.copyTo(src2final, gray);

    Mat finalImage = src1final + src2final;
    //Mat finalImage;
    //addWeighted(src1, 1, src2, 0, 0.0, finalImage);
    imwrite("C:\\Users\\ishaldin\\Desktop\\mycoupon.jpg", finalImage);
    namedWindow("output", WINDOW_AUTOSIZE);
    imshow("output", finalImage);
    cvWaitKey(0);

}

// Here we get four points from the user with left mouse clicks.
// On 5th click we output the overlayed image.
void on_mouse(int e, int x, int y, int d, void *ptr)
{
    if (e == EVENT_LBUTTONDOWN)
    {
        if (right_image.size() < 4)
        {

            right_image.push_back(Point2f(float(x), float(y)));
            cout << x << " " << y << endl;
        }
        else
        {
            cout << " Calculating Homography " << endl;
            // Deactivate callback
            cv::setMouseCallback("Display window second", NULL, NULL);
            destroyWindow("Display window second");
            // once we get 4 corresponding points in both images calculate homography matrix
            cout << "pppp" << left_image << "p-2" << right_image << endl;
            Mat H = findHomography(left_image, right_image, 0);
            Mat logoWarped;
            // Warp the logo image to change its perspective
            warpPerspective(imageLogo, logoWarped, H, imageMain.size());
            showFinal(imageMain, logoWarped);

        }

}


void on_mouse_left(int e, int x, int y, int d, void *ptr)
{
    if (e == EVENT_LBUTTONDOWN)
    {
        if (left_image.size() < 4)
        {

            left_image.push_back(Point2f(float(x), float(y)));
            cout << x << " " << y << endl;
        }
        else
        {
            cout << " Calculating Homography " << endl;
            // Deactivate callback
            cv::setMouseCallback("Display window first", NULL, NULL);
            destroyWindow("Display window first");
        }

    }
}

int main(int argc, char** argv)
{

    imageMain = imread("C:\\Users\\ishaldin\\Desktop\\1coupon.jpg", CV_LOAD_IMAGE_COLOR);

```



```



    imageLogo = imread("C:\\Users\\ishaldin\\Desktop\\2coupon.png", CV_LOAD_IMAGE_COLOR);
    // Push the 4 corners of the logo image as the 4 points for correspondence to calculate homography.




    namedWindow("Display window first", WINDOW_AUTOSIZE);// Create a window for display.
    imshow("Display window first", imageLogo);
    setMouseCallback("Display window first", on_mouse_left, NULL);


    imshow("Display window second", imageMain);


    setMouseCallback("Display window second", on_mouse, NULL);



    //  Press "Escape button" to exit
    while (1)
    {
        int key = cvWaitKey(10);
        if (key == 27) break;
    }


    return 0;
}
```