```
#include<iostream>
#include <stdio.h>
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>

//Debug Mode
#ifdef _DEBUG
#pragma comment(lib, "opencv_world330d.lib")
//Release Mode
#else
#pragma comment(lib, "opencv_world3430.lib")
#endif


int main(void)
{
    //グレースケール入力
    cv::Mat src = cv::imread("../image/diagram.bmp", cv::IMREAD_GRAYSCALE);

    //ラベリング処理
    cv::Mat LabelImg;
    cv::Mat stats;
    cv::Mat centroids;
    int nLab = cv::connectedComponentsWithStats(src, LabelImg, stats, centroids);

    // ラベリング結果の描画色を決定
    std::vector<cv::Vec3b> colors(nLab);
    colors[0] = cv::Vec3b(0, 0, 0);
    for (int i = 1; i < nLab; ++i) {
        colors[i] = cv::Vec3b((rand() & 255), (rand() & 255), (rand() & 255));
    }

    // ラベリング結果の描画
    cv::Mat Dst(src.size(), CV_8UC3);
    for (int i = 0; i < Dst.rows; ++i) {
        int *lb = LabelImg.ptr<int>(i);
        cv::Vec3b *pix = Dst.ptr<cv::Vec3b>(i);
        for (int j = 0; j < Dst.cols; ++j) {
            pix[j] = colors[lb[j]];
        }
    }

    //ROIの設定
    for (int i = 1; i < nLab; ++i) {
        int *param = stats.ptr<int>(i);

        int x = param[cv::ConnectedComponentsTypes::CC_STAT_LEFT];
        int y = param[cv::ConnectedComponentsTypes::CC_STAT_TOP];
        int height = param[cv::ConnectedComponentsTypes::CC_STAT_HEIGHT];
        int width = param[cv::ConnectedComponentsTypes::CC_STAT_WIDTH];

        cv::rectangle(Dst, cv::Rect(x, y, width, height), cv::Scalar(0, 255, 0), 2);
    }

    //重心の出力
    for (int i = 1; i < nLab; ++i) {
        double *param = centroids.ptr<double>(i);
        int x = static_cast<int>(param[0]);
        int y = static_cast<int>(param[1]);

        cv::circle(Dst, cv::Point(x, y), 3, cv::Scalar(0, 0, 255), -1);
    }

    //面積値の出力
    for (int i = 1; i < nLab; ++i) {
        int *param = stats.ptr<int>(i);
        std::cout << "area " << i << " = " << param[cv::ConnectedComponentsTypes::CC_STAT_AREA] <<
    std::endl;

        //ROIの左上に番号を書き込む
        int x = param[cv::ConnectedComponentsTypes::CC_STAT_LEFT];
        int y = param[cv::ConnectedComponentsTypes::CC_STAT_TOP];
```

```
        std::stringstream num;
        num << i;
        cv::putText(Dst, num.str(), cv::Point(x + 5, y + 20), cv::FONT_HERSHEY_COMPLEX, 0.7, cv::
    Scalar(0, 255, 255), 2);
    }

    cv::imshow("Src", src);
    cv::imshow("Labels", Dst);
    cv::waitKey();

    return 0;
}
```