1)

Como solução, penso na criação de uma segunda variável que teria um "limite" de 255. Ela funcionaria através de 1 "if", sendo a condição: "se [variável] > 255; então [variável] = 1". Assim, o contador de imagens continuaria aumentando e teria cor para colorir novas figuras (considerando que não seja um problema repetir cores para colorir).

Outra solução, talvez menos provável, seria colorir todas as figuras com apenas uma cor específica.

2)

```
#include <iostream>
#include <opencv2/opencv.hpp>

using namespace cv;
using namespace std;

int main(int argc, char** argv)
{
    cout << "Hello, Nairon!" << endl;

    Mat imagem, realce;
    imagem = imread("Imagens\\la.png", IMREAD_GRAYSCALE);
    if (!imagem.data) {
        std::cout << "\aErro ao abrir a imagem" << std::endl;
        return 1;
    }

    unsigned seed = time(0);
    srand(seed);
    int rdm = 1 + rand() % 1500;
    cout << "numero esperado de objetos (ignorando colisao) ---   " << rdm << "\n";
```

```
    for (int rd = 0; rd <rdm; rd++) {
        circle(imagem, Point(1 + rand() % imagem.rows, 1 + rand() % imagem.cols), 1,
Scalar(255), 2, 8);
    }

    imshow("Imagem com alguns objetos", imagem);

    Point p;
    p.x = 0;
    p.y = 0;
    int nObj = 0;
    int c = 0;
    for (int i = 0; i < imagem.rows; i++) {
        for (int j = 0; j < imagem.cols; j++) {
            if(imagem.at<uchar>(i, j) == 255){
                nObj++;
                c++;
                if (c == 256) {
                    c = 1;
                }
                p.x = j;
                p.y = i;
                floodFill(imagem, p, c);
            }
        }
    }

    imshow("Imagem final", imagem);
    cout << "Numero de objetos --- " << nObj << "\n\n\n";
    waitKey(0);
    return 0;
```

```
}
```