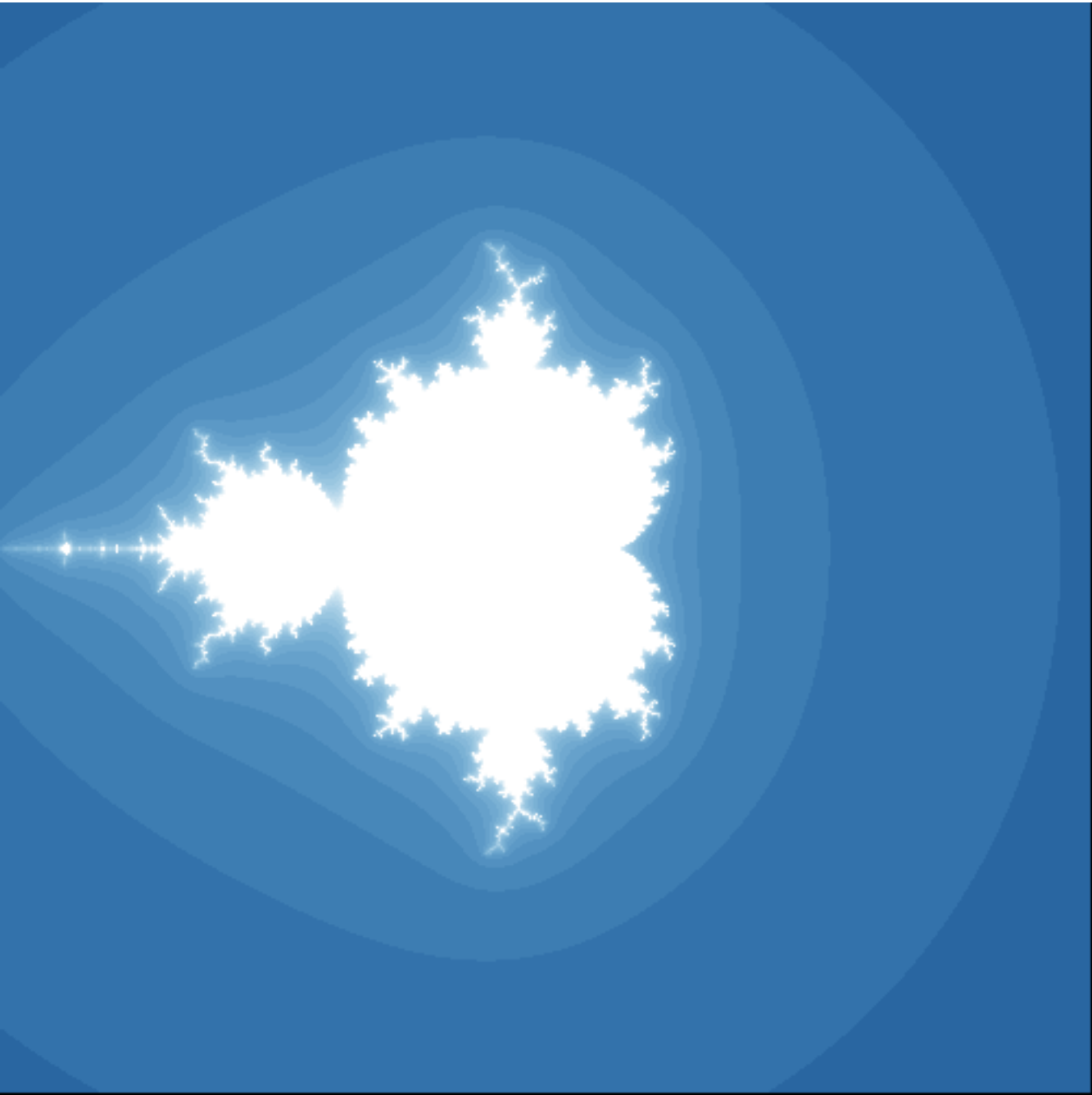




```

static void demoThree(PDF &p)
{
    // Create an image, 500 pixels square

    int width  = 500;
    int height = 500;

    Image anImage;
    RGB theColor(0, 0, 0);
    for(int i = 0; i < height; i ++)
    {
        ImageRow theRow;
        for(int j = 0; j < width; j ++)
            theRow.push_back(theColor);
        anImage.push_back(theRow);
    }

    double yStart = -2.0;
    double yStop  = 2.0;
    double yStep  = (yStop - yStart) / (height - 1);

    double xStart = -2.0;
    double xStop  = 2.0;
    double xStep  = (xStop - xStart) / (width - 1);

    int maxIterations = 25;
    double maxDistance = 1000.0;

    typedef complex<double> Complex;

    int iValue = 0;
    for(double y = yStart; y <= yStop; y += yStep)
    {
        int jValue = 0;
        for(double x = xStart; x <= xStop; x += xStep)
        {
            Complex z(0.0, 0.0);
            Complex c(x, y);

            int iterations = 0;
            while(
                iterations < maxIterations &&
                sqrt(z.real() * z.real() + z.imag() * z.imag()) < maxDistance
            )
            {
                z = z * z + c;
                iterations ++;
            }

            double v1 = (double)iterations / maxIterations;
            double v2 = sqrt(v1);
            double v3 = sqrt(v2);

            v1 *= 255.0;
            v2 *= 255.0;
            v3 *= 255.0;

            unsigned char red  = (unsigned char)(v1 + 0.5);
            unsigned char green = (unsigned char)(v2 + 0.5);
            unsigned char blue  = (unsigned char)(v3 + 0.5);

            RGB theColor(red, green, blue);

            anImage[iValue][jValue] = theColor;
            jValue++;
        }
        iValue++;
    }

    // Place the image, centered

    ImageInfo info = p.processImage(anImage);

    int xValue = (p.getWidth() / 2) - (width / 2);
    int yValue = (p.getHeight() / 2) - (height / 2);

    p.showImage(info, xValue, yValue, 1.0);

    p.newPage();
}

```

```
string errMsg;
vector<string> lines;

if(!getLines(__FILE__, lines, errMsg))
{
    cout << errMsg;
}
else
{
    static const int FONTSIZE = 8;
    static const int MARGIN   = 36;
    static const int YSTART   = 750;

    int y          = YSTART;
    bool showLine  = false;

    // Avoid false positive by buidling our
    // markerBegin and markerEnd strings up dynamically

    string tag      = "demoThree";
    string markerBegin = "// begin: " + tag;
    string markerEnd   = "// end: "   + tag;

    bool needSetFont = true;

    for(int i = 0, n = lines.size(); i < n; i++)
    {
        if(!showLine)
        {
            if(lines[i].find(markerBegin) != string::npos)
                showLine = true;
        }
        else
        {
            if(lines[i].find(markerEnd) != string::npos)
                showLine = false;
        }

        if(showLine)
        {
            if(needSetFont)
            {
                p.setFont(PDF::COURIER, FONTSIZE);
                needSetFont = false;
            }

            p.showTextXY(lines[i], MARGIN, y);
            y -= FONTSIZE;

            if(y <= MARGIN)
            {
                p.newPage();
                needSetFont = true;
                y = YSTART;
            }
        }
    }
}
}
```