```c
// implementation for mandelbrot set fractals
#include <graphics.h>
#include <stdio.h>
#define MAXCOUNT 30

// Function to draw mandelbrot set
void fractal(float left, float top, float xside, float yside)
{
    float xscale, yscale, zx, zy, cx, tempx, cy;
    int x, y, i, j;
    int maxx, maxy, count;

    // getting maximum value of x-axis of screen
    maxx = getmaxx();

    // getting maximum value of y-axis of screen
    maxy = getmaxy();

    // setting up the xscale and yscale
    xscale = xside / maxx;
    yscale = yside / maxy;

    // calling rectangle function
    // where required image will be seen
    rectangle(0, 0, maxx, maxy);

    // scanning every point in that rectangular area.
    // Each point represents a Complex number (x + yi).
    // Iterate that complex number
    for (y = 1; y <= maxy - 1; y++) {
        for (x = 1; x <= maxx - 1; x++)
        {
            // c_real
            cx = x * xscale + left;

            // c_imaginary
            cy = y * yscale + top;

            // z_real
            zx = 0;

            // z_imaginary
            zy = 0;
            count = 0;

            /* Calculate whether c(c_real + c_imaginary) belongs
             to the Mandelbrot set or not and draw a pixel
```

```
             at coordinates (x, y) accordingly
             If you reach the Maximum number of iterations
             and If the distance from the origin is
             greater than 2 exit the loop */
            while ((zx * zx + zy * zy < 4) && (count < MAXCOUNT))
            {
                // Calculate Mandelbrot function
                // z = z*z + c where z is a complex number

                // tempx = z_real*_real - z_imaginary*z_imaginary + c_real
                tempx = zx * zx - zy * zy + cx;

                // 2*z_real*z_imaginary + c_imaginary
                zy = 2 * zx * zy + cy;

                // Updating z_real = tempx
                zx = tempx;

                // Increment count
                count = count + 1;
            }

            // To display the created fractal
            putpixel(x, y, count);
        }
    }
}

// Driver code
int main()
{
    int gd = DETECT, gm, errorcode;
    char driver[]="";
    float left, top, xside, yside;

    // setting the left, top, xside and yside
    // for the screen and image to be displayed
    left = -1.75;
    top = -0.25;
    xside = 0.25;
    yside = 0.45;


    // initgraph initializes the
    // graphics system by loading a
    // graphics driver from disk
    initgraph(&gd, &gm, driver);

    // Function calling
```

```
    fractal(left, top, xside, yside);

    getch();

    // closegraph function closes the
    // graphics mode and deallocates
    // all memory allocated by
    // graphics system
    closegraph();

    return 0;
}
```