```
#include<iostream>
#include<graphics.h>
#include<algorithm>

using namespace std;

// Function to fill the polygon using Scan-Fill Algorithm
void scanFill(int vertices[][2], int n) {
    int maxY = 0;

    // Find the maximum y-coordinate of the polygon to define the scanline range
    for (int i = 0; i < n; i++) {
        maxY = max(maxY, vertices[i][1]);
    }

    // For each scanline (y from 0 to maxY), find the intersections with the polygon edges
    for (int y = 0; y <= maxY; y++) {
        // Create a list to store the x-coordinates of intersections for this scanline
        int intersections[100]; // Assuming no more than 100 intersections per scanline
        int intersectionCount = 0;

        // Loop through each edge and check if it intersects with the scanline
        for (int i = 0; i < n; i++) {
            int x1 = vertices[i][0];
            int y1 = vertices[i][1];
            int x2 = vertices[(i + 1) % n][0];
            int y2 = vertices[(i + 1) % n][1];

            // Check if the edge crosses the current scanline
            if ((y1 <= y && y2 > y) || (y2 <= y && y1 > y)) {
                // Calculate the x-coordinate of the intersection point
                int x_intersection = x1 + (y - y1) * (x2 - x1) / (y2 - y1);
                intersections[intersectionCount++] = x_intersection;
            }
        }

        // Sort the intersections in ascending order of x-coordinate
        sort(intersections, intersections + intersectionCount);

        // Fill the area between pairs of intersections
        for (int i = 0; i < intersectionCount; i += 2) {
            if (i + 1 < intersectionCount) {
                // Draw a horizontal line between each pair of intersections
                line(intersections[i], y, intersections[i + 1], y);
```

```
            }
        }
    }
}

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);

    // Define vertices of the polygon (a concave example)
    int vertices[4][2] = {{100, 100}, {50, 400}, {100, 200}, {150, 400}};

    // Draw the outline of the polygon
    for (int i = 0; i < 4; i++) {
        line(vertices[i][0], vertices[i][1], vertices[(i + 1) % 4][0], vertices[(i + 1) % 4][1]);
    }

    // Fill the polygon using Scan-Fill algorithm
    scanFill(vertices, 4);

    getch();
    closegraph();
    return 0;
}
```

Terminal commands:

```
g++ program_name.cpp -lgraph
./a.out
```