```
#include <iostream.h>
#include <conio.h>
#include <graphics.h>
#include <dos.h>
struct edge
{
  int x1, y1, x2, y2, flag;
};
int main() {
  int gd = DETECT, gm, n, i, j, k;
  struct edge ed[10], temped;
  float dx, dy, m[10], x_int[10], inter_x[10];
  int x[10], y[10], ymax = 0, ymin = 480, yy, temp;
  initgraph(&gd, &gm, "c:\\turboc3\\bgi");

  /*read the number of vertices of the polygon*/
  cout << "Enter the no.of vertices of the graph";
  cin >> n;
  /*read the vertices of the polygon and also find ymax and ymin*/
  cout << "Enter the vertices";
  for (i = 0; i < n; i++) {
    cin >> x[i] >> y[i];
    if (y[i] > ymax)
      ymax = y[i];
    if (y[i] < ymin)
      ymin = y[i];
    ed[i].x1 = x[i];
    ed[i].y1 = y[i];
  }

  /*store the edge information*/
  for (i = 0; i < n - 1; i++)  {
    ed[i].x2 = ed[i + 1].x1;
    ed[i].y2 = ed[i + 1].y1;
    ed[i].flag = 0;
  }
  ed[i].x2 = ed[0].x1;
  ed[i].y2 = ed[0].y1;
  ed[i].flag = 0;

  /*Check for y1>y2, if not interchnge y1 and y2 */
  for (i = 0; i < n; i++) {
    if (ed[i].y1 < ed[i].y2) {
      temp = ed[i].x1;
      ed[i].x1 = ed[i].x2; /////////////
      ed[i].x2 = temp;
      temp = ed[i].y1;
      ed[i].y1 = ed[i].y2;
      ed[i].y2 = temp;
    }
  }
```

```
/*Draw the polygon*/
for (i = 0; i < n; i++) {
  line(ed[i].x1, ed[i].y1, ed[i].x2, ed[i].y2);
}

/*sorting of edges in the order of y1,y2,x1*/
for (i = 0; i < n - 1; i++) {
  for (j = 0; j < n - 1; j++) {
    if (ed[j].y1 < ed[j + 1].y1) {
      temped = ed[j];
      ed[j] = ed[j + 1];
      ed[j + 1] = temped;
    }
    if (ed[j].y1 == ed[j + 1].y1) {
      if (ed[j].y2 < ed[j + 1].y2) {
        temped = ed[j];
        ed[j] = ed[j + 1];
        ed[j + 1] = temped;
      }
      if (ed[j].y2 == ed[j + 1].y2) {
        if (ed[j].x1 < ed[j + 1].x1) {
          temped = ed[j];
          ed[j] = ed[j + 1];
          ed[j + 1] = temped;
        }
      }
    }
  }
}

/*calculating 1/slope of each edge and storing top*/
for (i = 0; i < n; i++)  {
  dx = ed[i].x2 - ed[i].x1;
  dy = ed[i].y2 - ed[i].y1;
  if (dy == 0)
    m[i] = 0;
  else
    m[i] = dx / dy;
  inter_x[i] = ed[i].x1;
}

/*making the Actual edges*/
yy = ymax;
while (yy > ymin) {
  for (i = 0; i < n; i++) {
    if (yy > ed[i].y2 && yy <= ed[i].y1)
      ed[i].flag = 1;
    else
      ed[i].flag = 0;
  }
  j = 0;
```

```
    for (i = 0; i < n; i++) {
      if (ed[i].flag == 1) {
        if (yy == ed[i].y1) {
          x_int[j] = ed[i].x1;
          j++;
          if (ed[i - 1].y1 == yy && ed[i - 1].y1 < yy) {
            x_int[j] = ed[i].x1;
            j++;
          }
          if (ed[i + 1].y1 == yy && ed[i + 1].y1 < yy) {
            x_int[j] = ed[i].x1;
            j++;
          }
        }
        else {
          x_int[j] = inter_x[i] + (-m[i]);
          inter_x[i] = x_int[j];
          j++;
        }
      }
    }

    /*sorting the x intersaction*/
    for (i = 0; i < j; i++) {
      for (k = 0; k < j - 1; k++) {
        if (x_int[k] > x_int[k + 1]) {
          temp = (int)x_int[k];
          x_int[k] = x_int[k + 1];
          x_int[k + 1] = temp;
        }
      }
    }

    /*extracting pairs of values to draw lilnes*/
    for (i = 0; i < j; i = i + 2) {
      line((int)x_int[i], yy, (int)x_int[i + 1], yy);
    }
    yy--;
    delay(100);
  }
  getch();
  closegraph();
}
```