Practical 1

#include <conio.h>

#include <graphics.h>

int main() {

    int gd = DETECT, gm;

    initgraph(&gd, &gm,(char\*)"");

    putpixel(200, 200, RED);

    line(100, 100, 300, 300);

    setcolor(GREEN);

    circle(220, 220, 180);

    getch();

    closegraph();

    return 0;

}

Practical 2

#include <conio.h>

#include <graphics.h>

int main(){

    int gd = DETECT, gm;

    initgraph(&gd, &gm, (char\*)"");

    ellipse(300, 150, 0, 360, 100, 50);

    rectangle(100, 300, 300, 400);

    int points[] = {400, 200, 450, 150, 500, 200, 475, 250, 425, 250, 400, 200};

    drawpoly(6, points);

    int points2[] = {150, 100, 200, 50, 250, 100, 150, 100};

    drawpoly(4, points2);

    getch();

    closegraph();

    return 0;

}

Practical 3

#include <graphics.h>

#include <conio.h>

void drawLineDDA(int startX, int startY, int endX, int endY) {

    float currentX = startX, currentY = startY;

    int totalSteps = abs(endX-startX) > abs(endY-startY) ? abs(endX-startX) : abs(endY-startY);

    float xIncrement = (endX-startX)/(float)totalSteps;

    float yIncrement = (endY-startY)/(float)totalSteps;

    while(totalSteps-- >= 0) {

        putpixel(currentX, currentY, WHITE);

        currentX += xIncrement;

        currentY += yIncrement;

    }

}

int main() {

    int graphDriver = DETECT, graphMode;

    initgraph(&graphDriver, &graphMode, (char\*)"");

    drawLineDDA(100, 100, 400, 300);

    getch();

    closegraph();

    return 0;

}

Practical 4

#include <graphics.h>

#include <conio.h>

void drawLine(int x0, int y0, int x1, int y1) {

    int dx = abs(x1-x0), dy = abs(y1-y0);

    int sx = x0<x1 ? 1:-1, sy = y0<y1 ? 1:-1;

    int err = dx-dy;

    while(1) {

        putpixel(x0, y0, WHITE);

        if(x0==x1 && y0==y1) break;

        int e2 = 2\*err;

        if(e2 > -dy) {err -= dy; x0 += sx;}

        if(e2 < dx) {err += dx; y0 += sy;}

    }

}

int main() {

    int gd = DETECT, gm;

    initgraph(&gd, &gm, (char\*)"");

    drawLine(100, 100, 400, 300);

    getch();

    closegraph();

    return 0;

}

Practical 5

#include <graphics.h>

#include <conio.h>

#include <math.h>

void drawCircle(int x, int y, int r) {

    int cx = 0;

    int cy = r;

    while(cy >= cx) {

        putpixel(x + cx, y + cy, WHITE);

        putpixel(x - cx, y + cy, WHITE);

        putpixel(x + cx, y - cy, WHITE);

        putpixel(x - cx, y - cy, WHITE);

        putpixel(x + cy, y + cx, WHITE);

        putpixel(x - cy, y + cx, WHITE);

        putpixel(x + cy, y - cx, WHITE);

        putpixel(x - cy, y - cx, WHITE);

        cx++;

        cy = sqrt(r \* r - cx \* cx);

    }

}

int main() {

    int gd = DETECT, gm;

    initgraph(&gd, &gm, (char\*)"");

    drawCircle(250, 250, 200);

    getch();

    closegraph();

    return 0;

}

Practical 6

#include <graphics.h>

#include <stdio.h>

void fill(int x, int y, int fill, int border)

{

    if (getpixel(x, y) != border && getpixel(x, y) != fill)

    {

        putpixel(x, y, fill);

        fill(x + 1, y, fill, border);

        fill(x - 1, y, fill, border);

        fill(x, y + 1, fill, border);

        fill(x, y - 1, fill, border);

    }

}

int main()

{

    int driver = DETECT, mode;

    initgraph(&driver, &mode, (char \*)"");

    rectangle(25, 25, 275, 225);

    fill(26, 125, RED, WHITE);

    getch();

    closegraph();

    return 0;

}

Practical 8

#include <graphics.h>

#include <conio.h>

void tr(int p[][2], int tx, int ty, float s, int np) {

    for(int i=0; i<np; i++) {

        p[i][0] = p[i][0]\*s + tx;

        p[i][1] = p[i][1]\*s + ty;

    }

}

void dr(int p[][2], int np, int c) {

    setcolor(c);

    for(int i=0; i<np; i++)

        line(p[i][0], p[i][1], p[(i+1)%np][0], p[(i+1)%np][1]);

}

int main() {

    int gd=DETECT, gm;

    initgraph(&gd, &gm, (char\*)"");

    int sq[][2] = {{100,100}, {200,100}, {200,200}, {100,200}};

    dr(sq, 4, WHITE);

    tr(sq, 100, 100, 1, 4);

    dr(sq, 4, RED);

    tr(sq, 0, 0, 1.5, 4);

    dr(sq, 4, GREEN);

    getch();

    closegraph();

    return 0;

}

Practical 9

// 2D Rotation

#include <graphics.h>

#include <conio.h>

#include <cmath>

void drawPoly(int p[][2], int n)

{

    for (int v = 0; v < n; v++)

    {

        line(p[v][0], p[v][1],

             p[(v + 1) % n][0], p[(v + 1) % n][1]);

        line(p[v][0] + 1, p[v][1] + 1,

             p[(v + 1) % n][0] + 1, p[(v + 1) % n][1] + 1);

    }

}

void rot(int p[][2], int n, float a)

{

    float r = a \* M\_PI / 180;

    float c = cos(r);

    float s = sin(r);

    for (int v = 0; v < n; v++)

    {

        int x = p[v][0];

        int y = p[v][1];

        p[v][0] = round(x \* c - y \* s);

        p[v][1] = round(x \* s + y \* c);

    }

}

int main()

{

    int d = DETECT, m;

    initgraph(&d, &m, (char \*)"");

    int x = getmaxx() / 2;

    int y = getmaxy() / 2;

    int s[4][2] = {

        {x - 50, y - 50},

        {x + 50, y - 50},

        {x + 50, y + 50},

        {x - 50, y + 50}};

    setcolor(WHITE);

    drawPoly(s, 4);

    rot(s, 4, 30);

    setcolor(GREEN);

    drawPoly(s, 4);

    getch();

    closegraph();

    return 0;

}

Practical 10

#include <graphics.h>

#include <conio.h>

#include <cmath>

void dp(int p[][2], int n)

{

    for (int v = 0; v < n; v++)

        line(p[v][0], p[v][1], p[(v + 1) % n][0], p[(v + 1) % n][1]);

}

void rp(int p[][2], int n, int cx, int cy, float ad)

{

    float ar = ad \* M\_PI / 180;

    for (int v = 0; v < n; v++)

    {

        int rx = p[v][0] - cx;

        int ry = p[v][1] - cy;

        p[v][0] = rx \* cos(ar) - ry \* sin(ar) + cx;

        p[v][1] = rx \* sin(ar) + ry \* cos(ar) + cy;

    }

}

int main()

{

    int gd = DETECT, gm;

    initgraph(&gd, &gm, (char \*)"");

    int sq[4][2] = {{50, 50}, {350, 50}, {350, 350}, {50, 350}};

    setcolor(WHITE);

    dp(sq, 4);

    rp(sq, 4, 200, 200, 45);

    setcolor(GREEN);

    dp(sq, 4);

    getch();

    closegraph();

    return 0;

}

Practical 11

#include <graphics.h>

#include <math.h>

typedef struct

{

    float x, y, z;

} P3D;

void prj(P3D p, int \*sx, int \*sy)

{

    \*sx = p.x + p.z / 2;

    \*sy = p.y + p.z / 2;

}

void trns(P3D p[], int np, float tx, float ty, float tz, float s)

{

    for (int i = 0; i < np; i++)

    {

        p[i].x = p[i].x \* s + tx;

        p[i].y = p[i].y \* s + ty;

        p[i].z = p[i].z \* s + tz;

    }

}

void drCube(P3D v[])

{

    int sx[8], sy[8];

    for (int i = 0; i < 8; i++)

        prj(v[i], &sx[i], &sy[i]);

    for (int e = 0; e < 4; e++)

    {

        line(sx[e], sy[e], sx[(e + 1) % 4], sy[(e + 1) % 4]);

        line(sx[e + 4], sy[e + 4], sx[(e + 1) % 4 + 4], sy[(e + 1) % 4 + 4]);

        line(sx[e], sy[e], sx[e + 4], sy[e + 4]);

    }

}

int main()

{

    int gd = DETECT, gm;

    initgraph(&gd, &gm, (char \*)"");

    P3D cv[8] = {

        {25, 25, 100}, {125, 25, 100}, {125, 125, 100}, {25, 125, 100},

        {25, 25, 200}, {125, 25, 200}, {125, 125, 200}, {25, 125, 200}

    };

    setcolor(WHITE);

    drCube(cv);

    trns(cv, 8, 50, 50, 50, 1);

    setcolor(RED);

    drCube(cv);

    trns(cv, 8, 0, 0, 0, 1.5);

    setcolor(GREEN);

    drCube(cv);

    getch();

    closegraph();

    return 0;

}

Practical 13

#include <stdio.h>

#include <graphics.h>

#define INSIDE 0

#define LEFT 1

#define RIGHT 2

#define BOTTOM 4

#define TOP 8

int xmin, ymin, xmax, ymax;

int getCode(int x, int y)

{

    int code = INSIDE;

    if (x < xmin)

        code |= LEFT;

    else if (x > xmax)

        code |= RIGHT;

    if (y < ymin)

        code |= BOTTOM;

    else if (y > ymax)

        code |= TOP;

    return code;

}

void clip(int x1, int y1, int x2, int y2)

{

    int code1 = getCode(x1, y1), code2 = getCode(x2, y2);

    while (!(code1 == 0 && code2 == 0) && !(code1 & code2))

    {

        int ix, iy, outside = code1 ? code1 : code2;

        if (outside & TOP)

        {

            ix = x1 + (x2 - x1) \* (ymax - y1) / (y2 - y1);

            iy = ymax;

        }

        else if (outside & BOTTOM)

        {

            ix = x1 + (x2 - x1) \* (ymin - y1) / (y2 - y1);

            iy = ymin;

        }

        else if (outside & RIGHT)

        {

            iy = y1 + (y2 - y1) \* (xmax - x1) / (x2 - x1);

            ix = xmax;

        }

        else

        {

            iy = y1 + (y2 - y1) \* (xmin - x1) / (x2 - x1);

            ix = xmin;

        }

        if (outside == code1)

        {

            x1 = ix;

            y1 = iy;

            code1 = getCode(x1, y1);

        }

        else

        {

            x2 = ix;

            y2 = iy;

            code2 = getCode(x2, y2);

        }

    }

    if (code1 == 0 && code2 == 0)

    {

        line(x1, y1, x2, y2);

        printf("Line from (%d,%d) to (%d,%d)\n", x1, y1, x2, y2);

    }

}

int main()

{

    int gd = DETECT, gm;

    initgraph(&gd, &gm, NULL);

    xmin = 10;

    ymin = 10;

    xmax = 200;

    ymax = 200;

    rectangle(xmin, ymin, xmax, ymax);

    clip(5, 5, 150, 250);

    getch();

    closegraph();

    return 0;

}

Practical 15

#include <algorithm>

#include <graphics.h>

typedef struct

{

    int x, y;

} Pt;

int cwMinX = 10, cwMinY = 10, cwMaxX = 200, cwMaxY = 200;

void clipLine(Pt &p)

{

    p.x = std::max(cwMinX, std::min(p.x, cwMaxX));

    p.y = std::max(cwMinY, std::min(p.y, cwMaxY));

}

void drawPoly(Pt v[], int vc)

{

    for (int i = 0; i < vc; i++)

    {

        Pt p1 = v[i], p2 = v[(i + 1) % vc];

        clipLine(p1);

        clipLine(p2);

        line(p1.x, p1.y, p2.x, p2.y);

    }

}

int main()

{

    int gd = DETECT, gm;

    initgraph(&gd, &gm, NULL);

    Pt polyV[5] = {{50, 150}, {200, 50}, {300, 200}, {150, 250}, {50, 150}};

    int vc = 5;

    setcolor(WHITE);

    rectangle(cwMinX, cwMinY, cwMaxX, cwMaxY);

    setcolor(GREEN);

    drawPoly(polyV, vc);

    getch();

    closegraph();

    return 0;

}

Practical 16

#include <graphics.h>

void drawBezierCurve(int x[], int y[]) {

    for (float t = 0; t <= 1; t += 0.001) {

        float x0 = (1 - t) \* (1 - t) \* (1 - t) \* x[0] +

                  3 \* t \* (1 - t) \* (1 - t) \* x[1] +

                  3 \* t \* t \* (1 - t) \* x[2] +

                  t \* t \* t \* x[3];

        float y0 = (1 - t) \* (1 - t) \* (1 - t) \* y[0] +

                  3 \* t \* (1 - t) \* (1 - t) \* y[1] +

                  3 \* t \* t \* (1 - t) \* y[2] +

                  t \* t \* t \* y[3];

        putpixel(x0, y0, WHITE);

    }

}

int main() {

    int gd = DETECT, gm;

    initgraph(&gd, &gm, NULL);

    int x[] = {50, 200, 400, 550};

    int y[] = {300, 50, 550, 300};

    drawBezierCurve(x, y);

    getch();

    closegraph();

    return 0;

}