**Name: Deepankar Sharma course: BCA-6th roll no: 2092014**

**Subject: Computer Graphics**

|  |  |  |  |
| --- | --- | --- | --- |
| **Index** | | | |
| **S. No.** | **Objective** | **Date** | **Signature** |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |
| 10 |  |  |  |
| 11 |  |  |  |
| 12 |  |  |  |
| 13 |  |  |  |

NAME- Deepankar Sharma

COURSE- BCA

ROLL NO- 2092014

SUBJECT- Computer graphics lab

PRACTICLE-1

OBJECTIVE- DRAW A SMILEY FACE THOUGH GRAPHICS

SYNTAX :-

#include <graphics.h>

int main()

{

int gr = DETECT, gm;

initgraph(&gr, &gm, "C:\\Turboc3\\BGI");

setcolor(YELLOW);

circle(300, 100, 40);

setfillstyle(SOLID\_FILL, YELLOW);

floodfill(300, 100, YELLOW);

setcolor(BLACK);

setfillstyle(SOLID\_FILL, BLACK);

fillellipse(310, 85, 2, 6);

fillellipse(290, 85, 2, 6);

ellipse(300, 100, 205, 335, 20, 9);

ellipse(300, 100, 205, 335, 20, 10);

ellipse(300, 100, 205, 335, 20, 11);

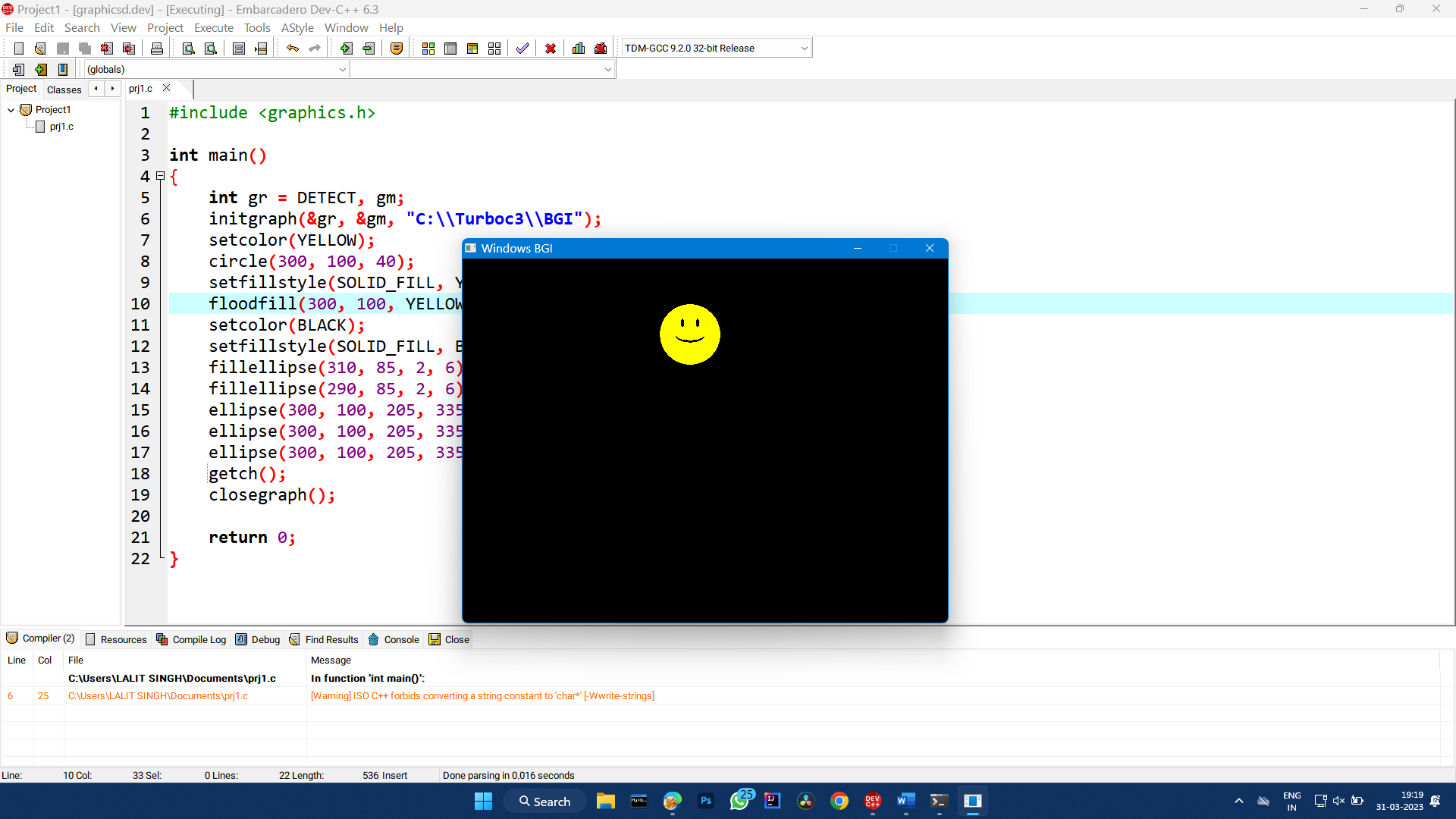
getch();

closegraph();

return 0;

}}

OUTPUT:

****

NAME- Deepankar Sharma

COURSE- BCA

ROLL NO- 2092014

SUBJECT- Computer graphics lab

PRACTICLE-2

OBJECTIVE- To divide your screen into four region, draw circle, rectangle, ellipse ,square.

SYNTAX :-

#include<conio.h>

#include<graphics.h>

#include<stdio.h>

int main()

{

int gdriver = DETECT, gmode;

int xmax,ymax;

initgraph(&gdriver, &gmode,"c:\\turboc3\\bgi");

xmax = getmaxx();

ymax = getmaxy();

line(xmax/2,0,xmax/2,ymax);

line(0,ymax/2,xmax,ymax/2);

outtextxy (xmax/2,ymax/2,"(0,0)");

setcolor(GREEN);

setfillstyle(HATCH\_FILL,RED);

circle(170,125,100);

outtextxy (160,135,"circle");

floodfill(170,125,GREEN);

setcolor(YELLOW);

setfillstyle(2,RED);

rectangle(58,251,304,392);

outtextxy (70,300,"Rectangle");

floodfill(70,351,YELLOW);

setcolor(BLUE);

setfillstyle(3,RED);

rectangle(400,50,500,150);

outtextxy (450,70,"square");

floodfill(450,80,BLUE);

setcolor(RED);

setfillstyle(4,RED);

ellipse(500,300,0,360,75,25);

outtextxy (500,300,"ellipse");

floodfill(500,300,RED);

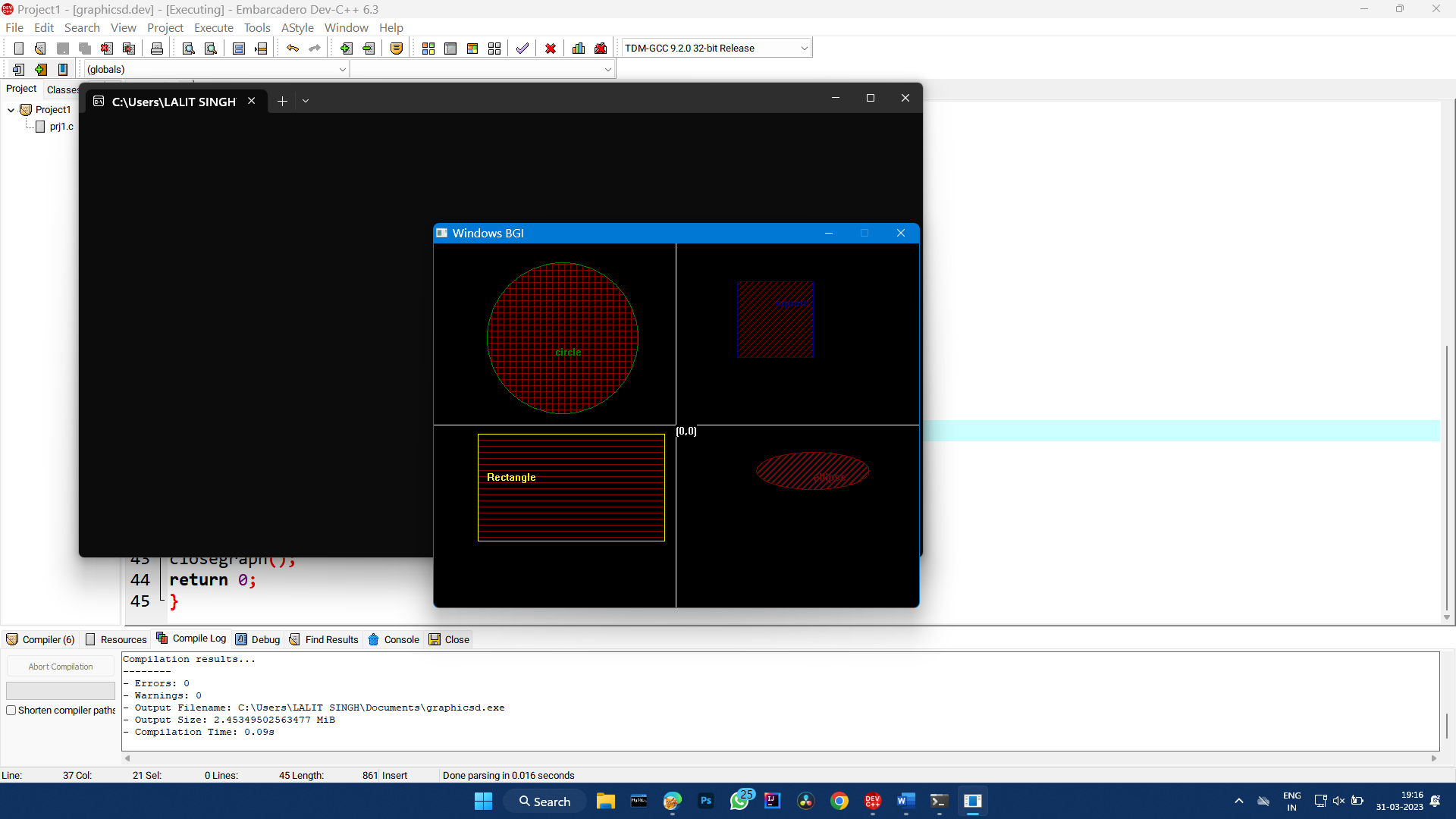
getch();

closegraph();

return 0;

}

OUTPUT:

****

NAME- Deepankar Sharma

COURSE- BCA

ROLL NO- 2092014

SUBJECT- Computer graphics lab

PRACTICLE- 3

OBJECTIVE- DRAW A HOUSE THOUGH GRAPHICS

SYNTAX :-

#include <graphics.h>

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, "");

rectangle(100, 200, 300, 400);

line(100, 200, 200, 100);

line(200, 100, 300, 200);

rectangle(120, 220, 180, 280);

rectangle(220, 220, 280, 280);

rectangle(160, 320, 240, 400);

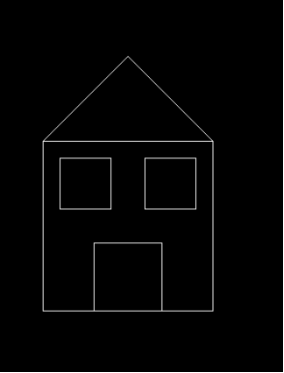
getch();

closegraph();

return 0;

}

OUTPUT:



NAME- Deepankar Sharma

COURSE- BCA

ROLL NO- 2092014

SUBJECT- Computer graphics lab

PRACTICLE-4

OBJECTIVE- TO IMPLEMENT THE DDA LINE GENERATION ALGORITHM THOUGH GRAPHICS

SYNTAX :-

#include<graphics.h>

#include<conio.h>

#include<stdio.h>

int main()

{

int gd = DETECT ,gm, i;

float x, y,dx,dy,steps;

int x0, x1, y0, y1;

initgraph(&gd, &gm, "C:\\TC\\BGI");

x0 = 200 , y0 = 300, x1 = 500, y1 = 100;

dx = (float)(x1 - x0);

dy = (float)(y1 - y0);

if(dx>=dy)

{

steps = dx;

}

else

{

steps = dy;

}

dx = dx/steps;

dy = dy/steps;

x = x0;

y = y0;

i = 1;

while(i<= steps)

{

putpixel(x, y, WHITE);

x += dx;

y += dy;

i=i+1;

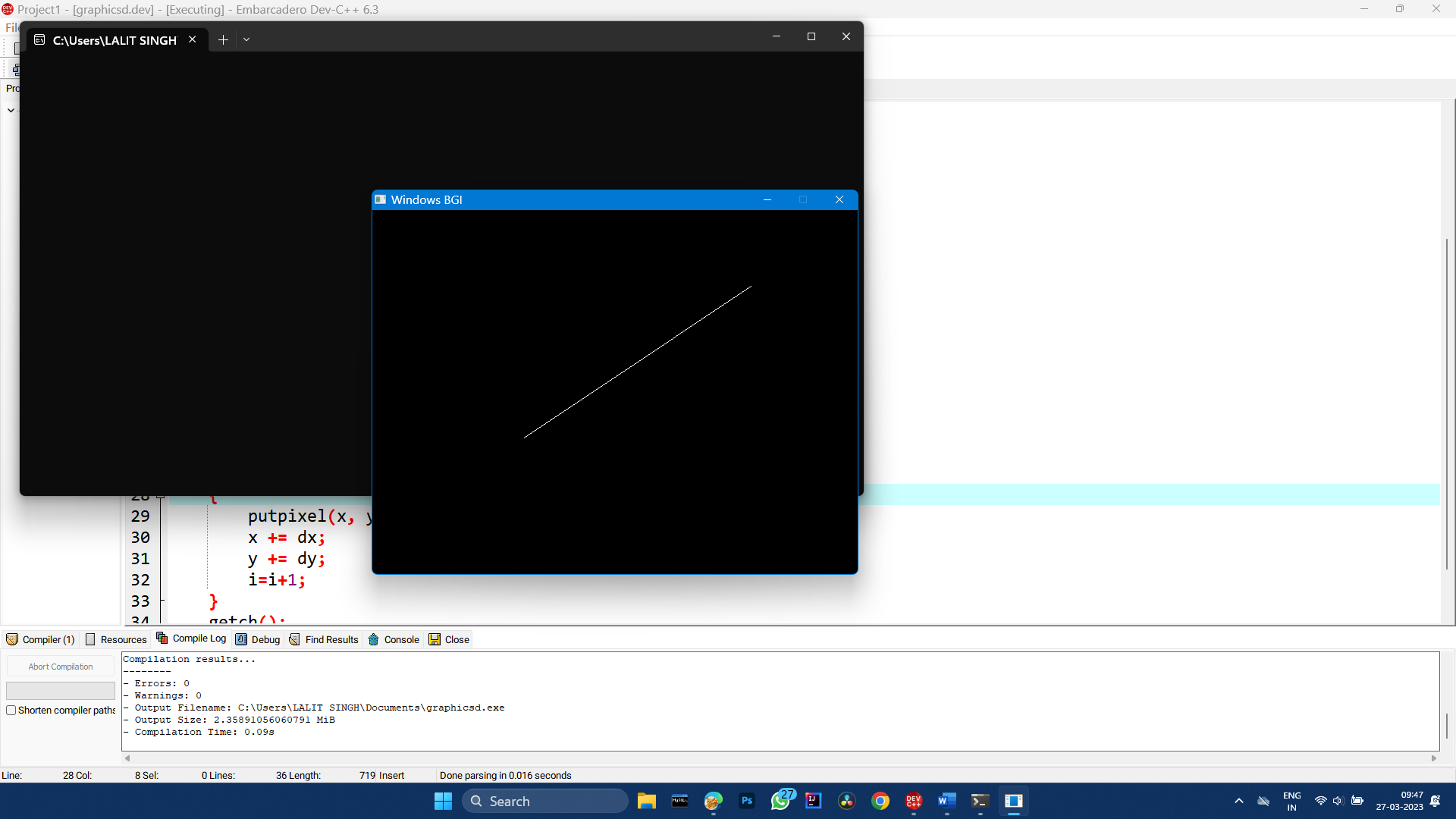
}

getch();

closegraph();

}

OUTPUT:

****

NAME- Deepankar Sharma

COURSE- BCA

ROLL NO- 2092014

SUBJECT- Computer graphics lab

PRACTICLE-5

OBJECTIVE- TO IMPLEMENT THE Bresenham's Line Algorithm THOUGH GRAPHICS

SYNTAX :-

#include <iostream>

#include <graphics.h>

void bresenham(int x1, int y1, int x2, int y2) {

int dx = x2 - x1;

int dy = y2 - y1;

int p = 2 \* dy - dx;

int twoDy = 2 \* dy;

int twoDyMinusDx = 2 \* (dy - dx);

int x = x1;

int y = y1;

if (x1 > x2) {

x = x2;

y = y2;

x2 = x1;

} else {

x = x1;

y = y1;

}

putpixel(x, y, WHITE);

while (x < x2) {

x++;

if (p < 0) {

p += twoDy;

} else {

y++;

p += twoDyMinusDx;

}

putpixel(x, y, BLUE);

}

}

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, "");

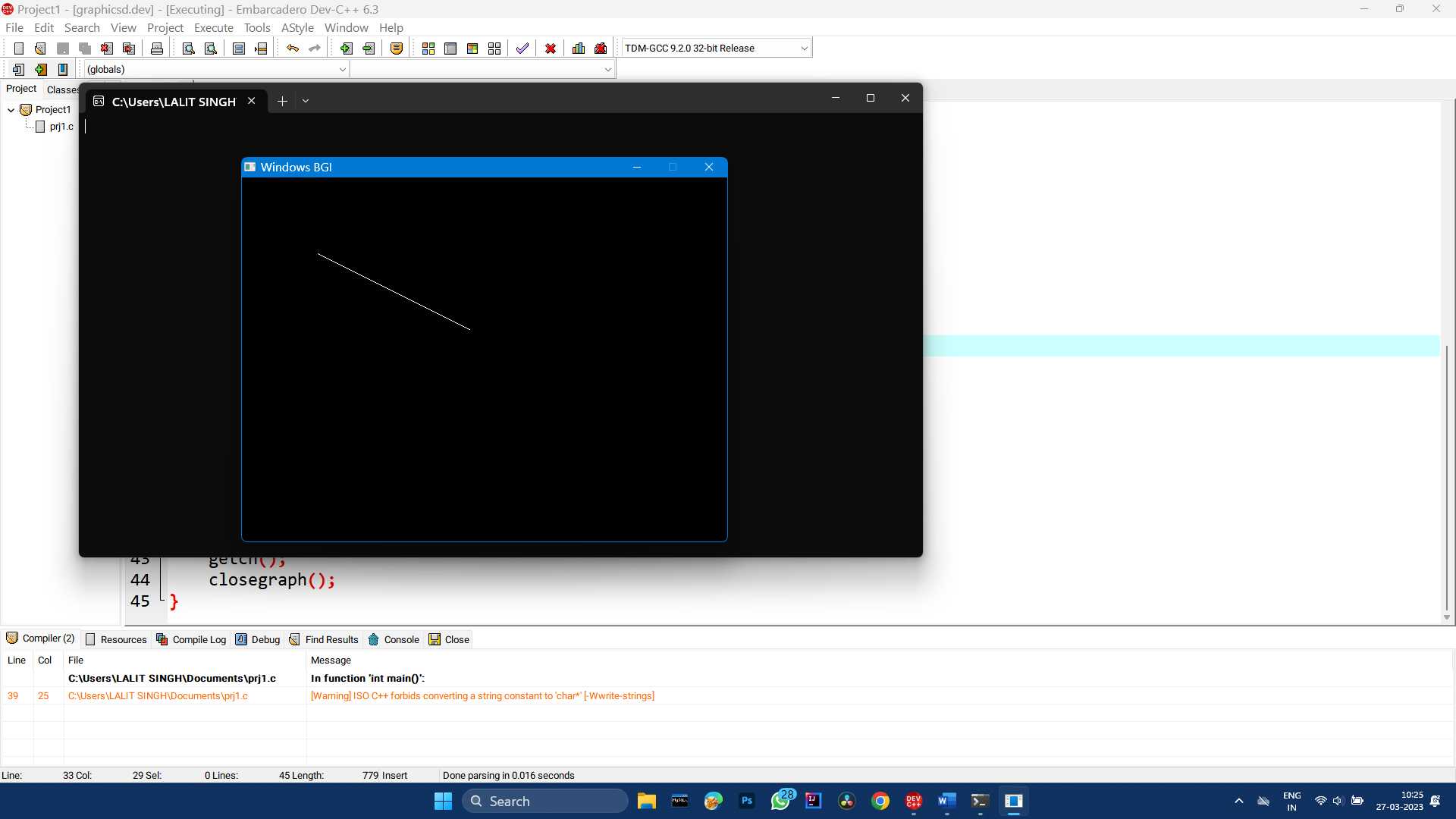
bresenham(100, 100, 300, 200);

getch();

closegraph();

}

OUTPUT:

****

NAME- Deepankar Sharma

COURSE- BCA

ROLL NO- 2092014

SUBJECT- Computer graphics lab

PRACTICLE-6

OBJECTIVE- To implement Mid Point Circle drawing Algorithm through graphics.

SYNTAX:-

#include<graphics.h>

#include<iostream>

using namespace std;

// Midpoint Circle drawing Algorithm

void drawMidPointCircle(int x0, int y0, int radius)

{

int x = radius, y = 0;

int decisionParam = 1 - radius;

while (y <= x)

{

putpixel(x0 + x, y0 + y, 1);

putpixel(x0 - x, y0 + y, 2);

putpixel(x0 + x, y0 - y, 3);

putpixel(x0 - x, y0 - y, 4);

putpixel(x0 + y, y0 + x, 5);

putpixel(x0 - y, y0 + x, 6);

putpixel(x0 + y, y0 - x, 7);

putpixel(x0 - y, y0 - x, 8);

y++;

if (decisionParam <= 0)

decisionParam += 2 \* y + 1;

else

{

x--;

decisionParam += 2 \* (y - x) + 1;

}

}

}

int main()

{

int gDrive = DETECT;

int gMode;

initgraph(&gDrive, &gMode, NULL);

int X0 = 0, Y0 = 0, radius=0 ;

printf("The constraint on the X-axis are(0-%d)\n", getmaxx());

printf("The constraint on the Y-axis are(0-%d)\n", getmaxy());

cout<<("Enter the X0: ");

scanf("%d", &X0);

cout<<("Enter the Y0: ");

scanf("%d", &Y0);

cout<<("Enter the radius: ");

scanf("%d", &radius);

// Function call

// DDA(X0, Y0, X1, Y1);

drawMidPointCircle(X0, Y0, radius);

// DDA(2, 2, 14, 16);

getch();

closegraph();

return 0;

}

OUTPUT:-

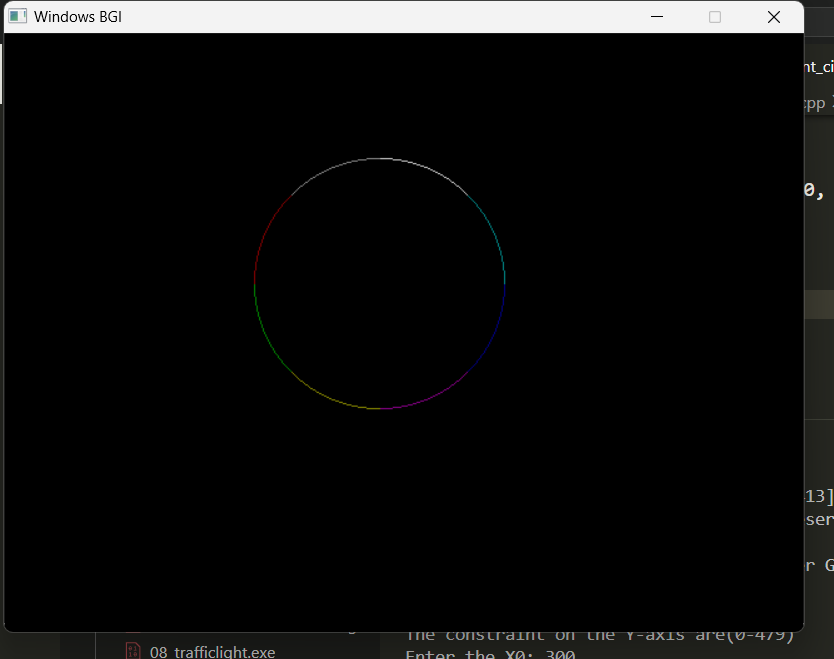
The constraint on the X-axis are(0-639)

The constraint on the Y-axis are(0-479)

Enter the X0: 300

Enter the Y0: 200

Enter the radius: 100



NAME- Deepankar Sharma

COURSE- BCA

ROLL NO- 2092014

SUBJECT- Computer graphics lab

PRACTICLE-7

OBJECTIVE- To implement Brensanham’s Circle drawing Algorithm through graphics.

SYNTAX:-

#include <graphics.h>

#include <iostream>

using namespace std;

// Midpoint Circle drawing Algorithm

void drawMidPointCircle(int x0, int y0, int radius)

{

int x = radius, y = 0;

int decisionParam = 1 - radius;

while (y <= x)

{

putpixel(x0 + x, y0 + y, 1);

putpixel(x0 - x, y0 + y, 2);

putpixel(x0 + x, y0 - y, 3);

putpixel(x0 - x, y0 - y, 4);

putpixel(x0 + y, y0 + x, 5);

putpixel(x0 - y, y0 + x, 6);

putpixel(x0 + y, y0 - x, 7);

putpixel(x0 - y, y0 - x, 8);

y++;

if (decisionParam <= 0)

decisionParam += 2 \* y + 1;

else

{

x--;

decisionParam += 2 \* (y - x) + 1;

}

}

}

// Brensanham Circle drawing Algorithm

void drawBrensanhamCircle(int x0, int y0, int radius)

{

int x = 0, y = radius;

int decisionParam = 3 - 2 \* radius;

while (x <= y)

{

putpixel(x0 + x, y0 + y, RED);

putpixel(x0 + y, y0 + x, RED);

putpixel(x0 - y, y0 + x, RED);

putpixel(x0 - x, y0 + y, RED);

putpixel(x0 - x, y0 - y, RED);

putpixel(x0 - y, y0 - x, RED);

putpixel(x0 + y, y0 - x, RED);

putpixel(x0 + x, y0 - y, RED);

if (decisionParam <= 0)

{

x++;

decisionParam += 4 \* x + 6;

}

else

{

x++;

y--;

decisionParam += 4 \* (x - y) + 10;

}

}

}

int main()

{

int gDrive = DETECT;

int gMode;

initgraph(&gDrive, &gMode, NULL);

int X0 = 0, Y0 = 0, radius = 0;

printf("The constraint on the X-axis are(0-%d)\n", getmaxx());

printf("The constraint on the Y-axis are(0-%d)\n", getmaxy());

cout << ("Enter the X0: ");

scanf("%d", &X0);

cout << ("Enter the Y0: ");

scanf("%d", &Y0);

cout << ("Enter the radius: ");

scanf("%d", &radius);

// Function call

// DDA(X0, Y0, X1, Y1);

drawBrensanhamCircle(X0, Y0, radius);

// DDA(2, 2, 14, 16);

getch();

closegraph();

return 0;

}

OUTPUT:-

C:\Deepankar\06\_semester\TBC 601 Computer Graphics\PracticalsVScode>"c:\Deepankar\06\_semester\TBC 601 Computer Graphics\PracticalsVScode\Home\build\07\_Bresenham\_circleDrawing.exe"

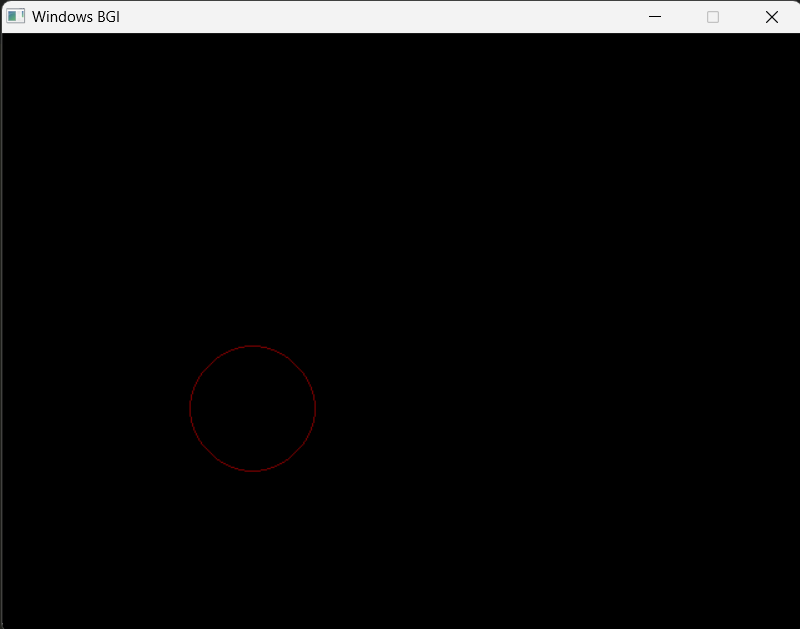
The constraint on the X-axis are(0-639)

The constraint on the Y-axis are(0-479)

Enter the X0: 200

Enter the Y0: 300

Enter the radius: 50



NAME- Deepankar Sharma

COURSE- BCA

ROLL NO- 2092014

SUBJECT- Computer graphics lab

PRACTICLE-8

OBJECTIVE- To implement Blinking Traffic Light through graphics.

SYNTAX:-

#include <iostream>

#include <graphics.h>

using namespace std;

void drawTrafficLight(int x, int y, int lightSize, bool isRedOn, bool isYellowOn, bool isGreenOn)

{

// Draw black background

setfillstyle(SOLID\_FILL, BLACK);

bar(x, y, x + lightSize, y + 3 \* lightSize);

// Draw red light

setfillstyle(SOLID\_FILL, isRedOn ? RED : DARKGRAY);

circle(x + lightSize / 2, y + lightSize / 2, lightSize / 2);

floodfill(x + lightSize / 2, y + lightSize / 2, WHITE);

// Draw yellow light

setfillstyle(SOLID\_FILL, isYellowOn ? YELLOW : DARKGRAY);

circle(x + lightSize / 2, y + lightSize + lightSize / 2, lightSize / 2);

floodfill(x + lightSize / 2, y + lightSize + lightSize / 2, WHITE);

// Draw green light

setfillstyle(SOLID\_FILL, isGreenOn ? GREEN : DARKGRAY);

circle(x + lightSize / 2, y + 2 \* lightSize + lightSize / 2, lightSize / 2);

floodfill(x + lightSize / 2, y + 2 \* lightSize + lightSize / 2, WHITE);

}

int main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "");

int lightSize = 100;

int x = (getmaxx() - lightSize) / 2;

int y = (getmaxy() - 3 \* lightSize) / 2;

while (true)

{

drawTrafficLight(x, y, lightSize, true, false, false);

delay(400);

drawTrafficLight(x, y, lightSize, true, true, false);

delay(400);

drawTrafficLight(x, y, lightSize, false, false, true);

delay(400);

drawTrafficLight(x, y, lightSize, false, true, false);

delay(400);

}

getch();

closegraph();

return 0;

}

OUTPUT:-

