Program-1: Print Pixel

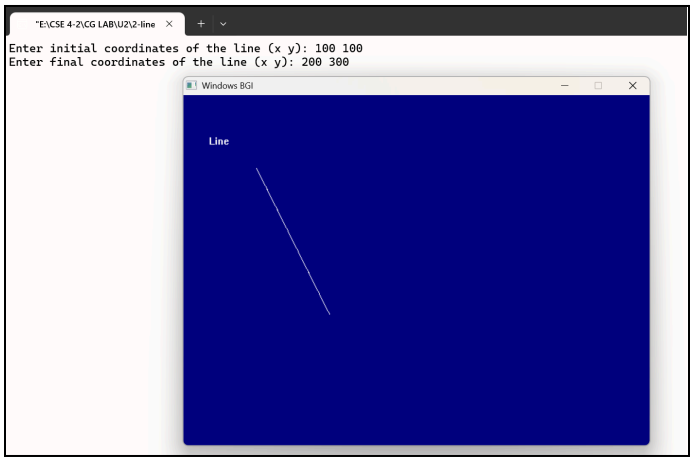
```
#include <graphics.h>
#include <conio.h>
#include <dos.h>
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
    setbkcolor(BLUE);
    cleardevice();
    putpixel(50, 100, YELLOW);
    outtextxy(35, 55, (char *)"PIXEL");
    getch();
    closegraph();
    return 0;
}
```

Output-1:

2.Program – 2: Print line using line()

```
#include <graphics.h>
#include <conio.h>
#include <iostream>
using namespace std;
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
    int x_initial, y_initial, x_final, y_final;
    cout << "Enter initial coordinates of the line (x y): ";
    cin >> x_initial >> y_initial;
    cout << "Enter final coordinates of the line (x y): ";
    cin >> x_final >> y_final;
    setbkcolor(BLUE);
    cleardevice();
    outtextxy(35, 55, (char *)"Line");
    line(x_initial, y_initial, x_final, y_final);
    getch();
    closegraph();
    return 0;
}
```

Output-2:

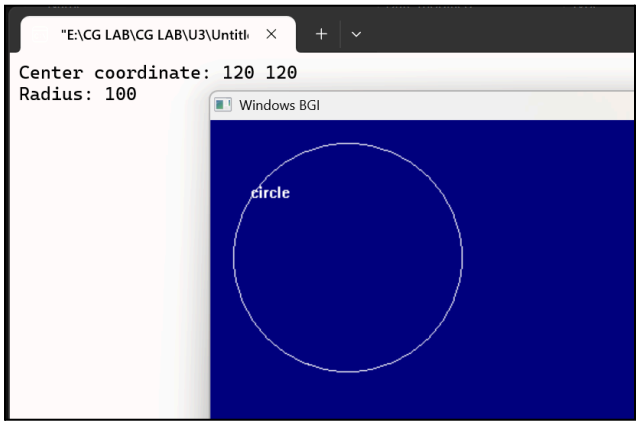


Program – 3: Print circle using circle()

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    int h, k, radius;
    printf("Center coordinate: ");
    scanf("%d %d", &h, &k);
    printf("Radius: ");
    scanf("%d", &radius);
    setbkcolor(BLUE);
    cleardevice();
    outtextxy(35, 55, "circle");
    circle(h, k, radius);
    getch();
    closegraph();
    return 0;
}
```

Output-3:

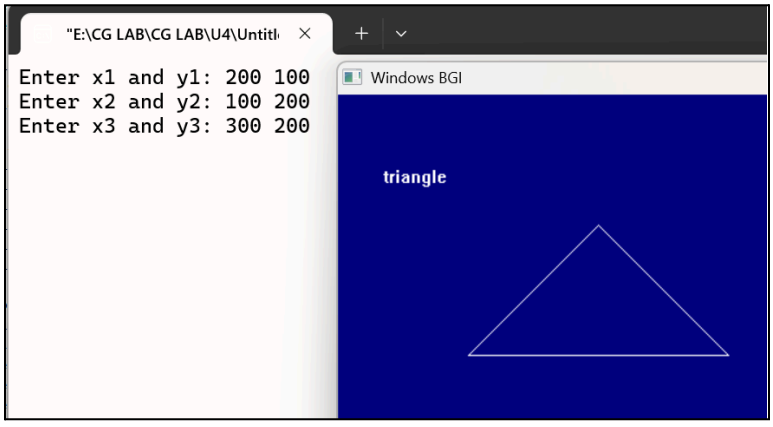


Program – 4: print triangle using line()

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    int x1, y1, x2, y2, x3, y3;
    printf("Enter x1 and y1: ");
    scanf("%d %d", &x1, &y1);
    printf("Enter x2 and y2: ");
    scanf("%d %d", &x2, &y2);
    printf("Enter x3 and y3: ");
    scanf("%d %d", &x3, &y3);
    setbkcolor(BLUE);
    cleardevice();
    outtextxy(35, 55, "triangle");
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
    getch();
    closegraph();
    return 0;
}
```

Output-4:

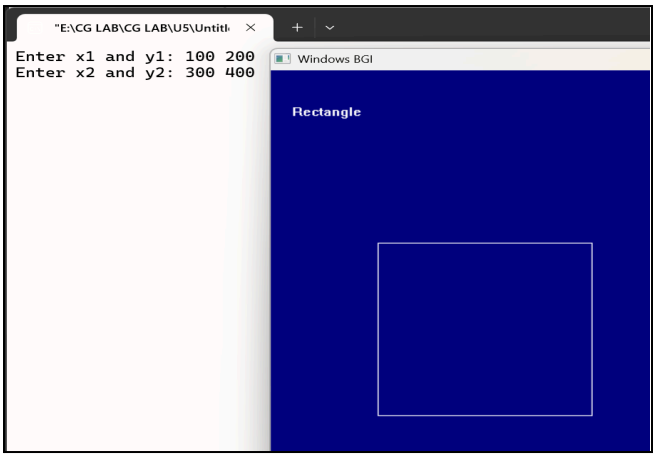


Program – 5: Draw rectangle using rectangle()

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    int x1, y1, x2, y2;
    printf("Enter x1 and y1: ");
    scanf("%d %d", &x1, &y1);
    printf("Enter x2 and y2: ");
    scanf("%d %d", &x2, &y2);
    setbkcolor(BLUE);
    cleardevice();
    outtextxy(35, 55, "Rectangle");
    rectangle(x1, y1, x2, y2);
    getch();
    closegraph();
    return 0;
}
```

Output-5:

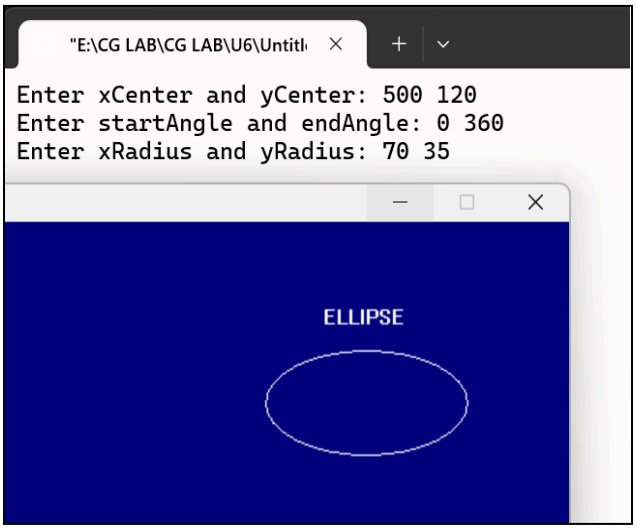


Program – 6: Draw ellipse using ellipse()

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    int x1, y1, x2, y2, x3, y3;
    printf("Enter xCenter and yCenter: ");
    scanf("%d %d", &x1, &y1);
    printf("Enter startAngle and endAngle: ");
    scanf("%d %d", &x2, &y2);
    printf("Enter xRadius and yRadius: ");
    scanf("%d %d", &x3, &y3);
    setbkcolor(BLUE);
    cleardevice();
    outtextxy(470, 55, "ELLIPSE");
    ellipse(x1, y1, x2, y2, x3, y3);
    getch();
    closegraph();
    return 0;
}
```

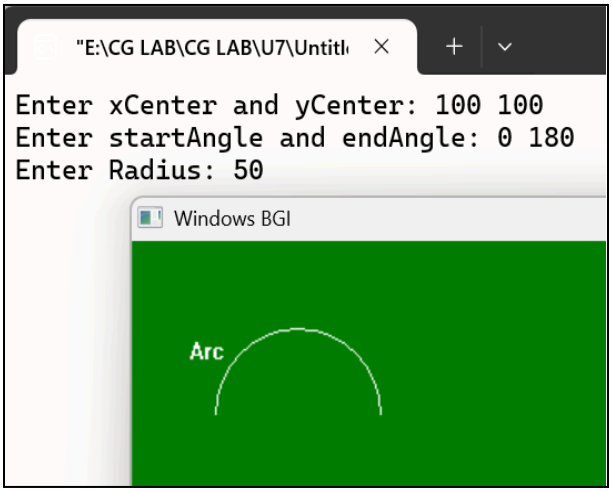
Output-6:



Program – 7: Draw Arc using arc()

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    int x1, y1, x2, y2, r;
    printf("Enter xCenter and yCenter: ");
    scanf("%d %d", &x1, &y1);
    printf("Enter startAngle and endAngle: ");
    scanf("%d %d", &x2, &y2);
    printf("Enter Radius: ");
    scanf("%d", &r);
    setbkcolor(GREEN);
    cleardevice();
    outtextxy(35, 55, "Arc");
    arc(x1, y1, x2, y2, r);
    getch();
    closegraph();
    return 0;
}
```

Output-7:

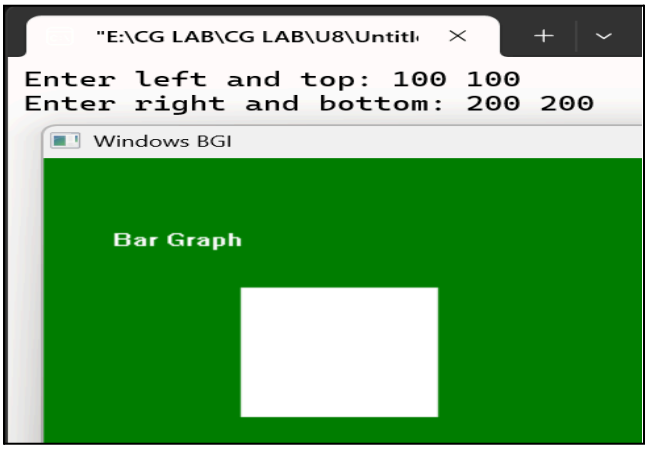


Program – 8: Draw a bar using bar()

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    int x1, y1, x2, y2;
    printf("Enter left and top: ");
    scanf("%d %d", &x1, &y1);
    printf("Enter right and bottom: ");
    scanf("%d %d", &x2, &y2);
    setbkcolor(GREEN);
    cleardevice();
    outtextxy(35, 55, "Bar Graph");
    bar(x1, y1, x2, y2);
    getch();
    closegraph();
    return 0;
}
```

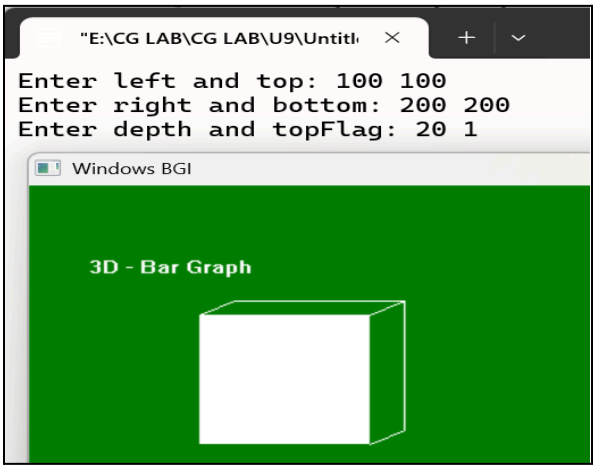
Output-8:



Program – 9 : Draw 3D bar using bar3d()

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");
    int x1, y1, x2, y2, depth, topFlag;
    printf("Enter left and top: ");
    scanf("%d %d", &x1, &y1);
    printf("Enter right and bottom: ");
    scanf("%d %d", &x2, &y2);
    printf("Enter depth and topFlag: ");
    scanf("%d %d", &depth, &topFlag);
    setbkcolor(GREEN);
    cleardevice();
    outtextxy(35, 55, (char *)"3D - Bar Graph");
    bar3d(x1, y1, x2, y2, depth, topFlag);
    getch();
    closegraph();
    return 0;
}
```

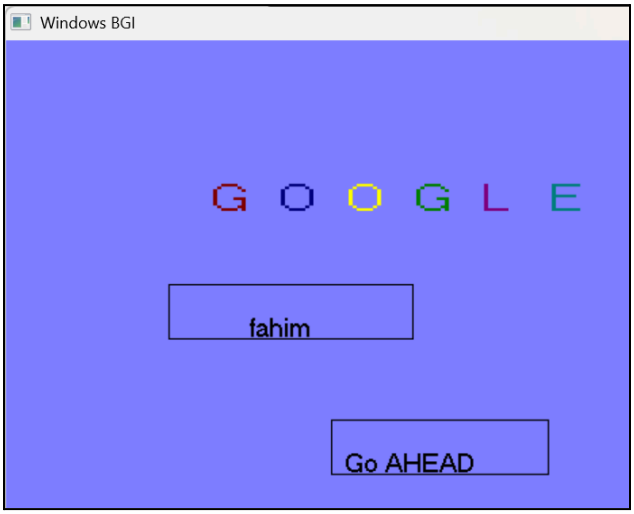
Output-9:



Program – 10: Draw a Home Page

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
    setbkcolor(LIGHTBLUE);
    cleardevice();
    setcolor(RED);
    settextstyle(SANS_SERIF_FONT, HORIZ_DIR, 4);
    outtextxy(150, 100, (char *)"G");
    setcolor(BLUE);
    outtextxy(200, 100, (char *)"O");
    setcolor(YELLOW);
    outtextxy(250, 100, (char *)"O");
    setcolor(GREEN);
    outtextxy(300, 100, (char *)"G");
    setcolor(MAGENTA);
    outtextxy(350, 100, (char *)"L");
    setcolor(CYAN);
    outtextxy(400, 100, (char *)"E");
    setcolor(BLACK);
    settextstyle(SANS_SERIF_FONT, HORIZ_DIR, 2);
    outtextxy(180, 200, (char *)"fahim");
    outtextxy(250, 300, (char *)"Go AHEAD");
    rectangle(120, 180, 300, 220);
    rectangle(240, 280, 400, 320);
    getch();
    closegraph();
    return 0;
}
```

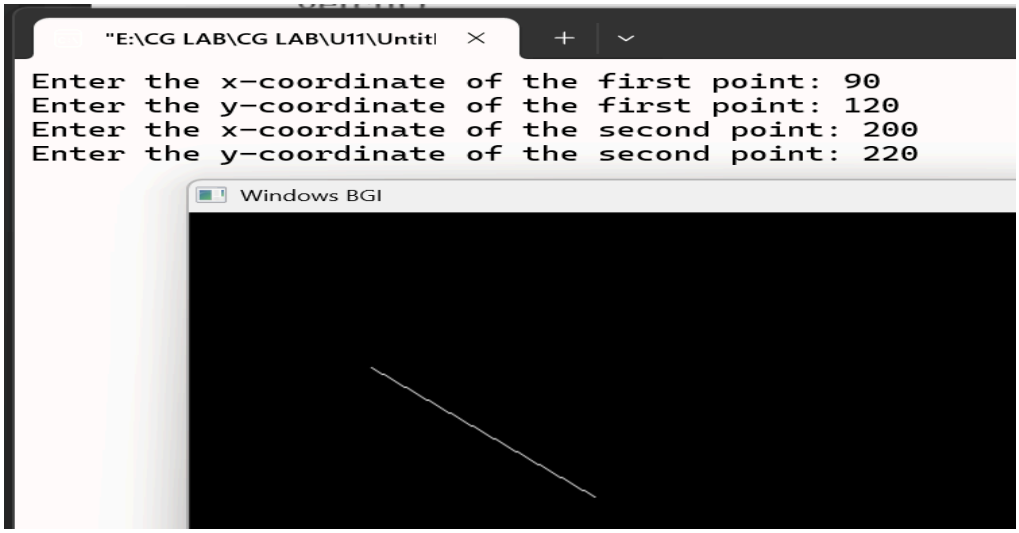
Output-10:



Program – 11: Write a C++ program to implement Bresenham's Line Drawing Algorithm.

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
int main() {
    int x, y, x1, y1, x2, y2, p, dx, dy;
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TurboC3\\BGI");
    printf("Enter the x-coordinate of the first point: ");
    scanf("%d", &x1);
    printf("Enter the y-coordinate of the first point: ");
    scanf("%d", &y1);
    printf("Enter the x-coordinate of the second point: ");
    scanf("%d", &x2);
    printf("Enter the y-coordinate of the second point: ");
    scanf("%d", &y2);
    x = x1;
    y = y1;
    dx = x2 - x1;
    dy = y2 - y1;
    p = 2 * dy - dx;
    putpixel(x, y, WHITE);
    while (x < x2) {
        x++;
        if (p < 0) {
            p = p + 2 * dy;
        } else {
            y++;
            p = p + 2 * dy - 2 * dx;
        }
        putpixel(x, y, WHITE);
    }
    getch();
    closegraph();
    return 0;
}
```

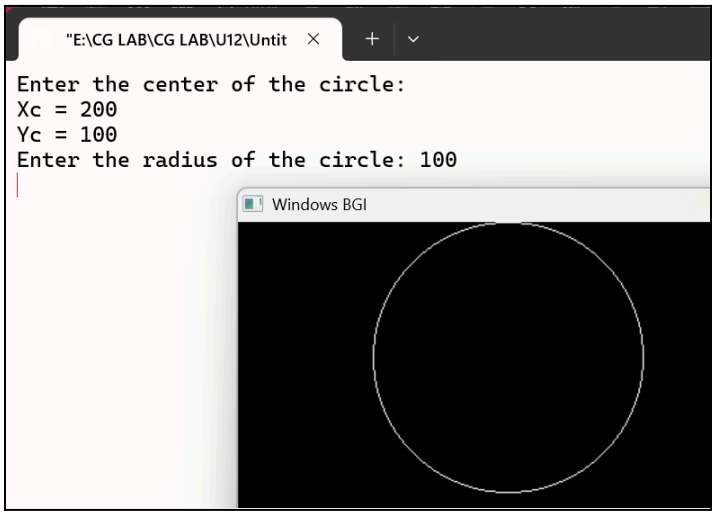
Output-11:



Program–12: Develop a C++ program to implement the Midpoint Circle Drawing Algorithm.

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
void draw_circle(int, int, int);
void symmetry(int, int, int, int);
int main() {
    int xc, yc, R;
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TurboC3\\BGI");
    printf("Enter the center of the circle:\n");
    printf("Xc = ");
    scanf("%d", &xc);
    printf("Yc = ");
    scanf("%d", &yc);
    printf("Enter the radius of the circle: ");
    scanf("%d", &R);
    draw_circle(xc, yc, R);
    getch();
    closegraph();
    return 0;
}
void draw_circle(int xc, int yc, int rad) {
    int x = 0;
    int y = rad;
    int p = 1 - rad;
    symmetry(x, y, xc, yc);
    while (x < y) {
        x++;
        if (p < 0) {
            p += 2 * x + 1;
        } else {
            y--;
            p += 2 * (x - y) + 1;
        }
        symmetry(x, y, xc, yc);
        delay(50);
    }
}
void symmetry(int x, int y, int xc, int yc) {
    putpixel(xc + x, yc + y, GREEN);
    putpixel(xc - x, yc + y, GREEN);
    putpixel(xc + x, yc - y, GREEN);
    putpixel(xc - x, yc - y, GREEN);
    putpixel(xc + y, yc + x, GREEN);
    putpixel(xc - y, yc + x, GREEN);
    putpixel(xc + y, yc - x, GREEN);
    putpixel(xc - y, yc - x, GREEN);
}
```

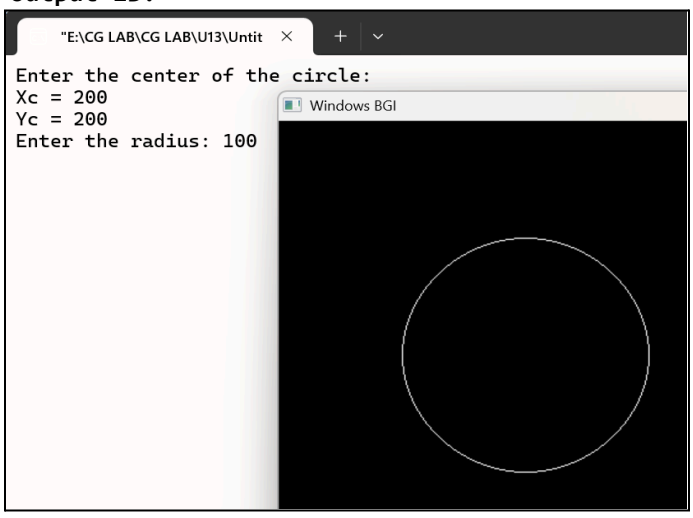
Output-12:



Program – 13: Develop a C++ program to implement the Bresenham's Circle Drawing Algorithm.

```
#include <graphics.h>
#include <dos.h>
#include <stdio.h>
void drawCircle(int xc, int yc, int x, int y) {
    putpixel(xc + x, yc + y, WHITE);
    putpixel(xc - x, yc + y, WHITE);
    putpixel(xc + x, yc - y, WHITE);
    putpixel(xc - x, yc - y, WHITE);
    putpixel(xc + y, yc + x, WHITE);
    putpixel(xc - y, yc + x, WHITE);
    putpixel(xc + y, yc - x, WHITE);
    putpixel(xc - y, yc - x, WHITE);
}
void circleBres(int xc, int yc, int r) {
    int x = 0, y = r;
    int d = 3 - 2 * r;
    drawCircle(xc, yc, x, y);
    while (y >= x) {
        x++;
        if (d > 0) {
            y--;
            d = d + 4 * (x - y) + 10;
        } else {
            d = d + 4 * x + 6;
        }
        drawCircle(xc, yc, x, y);
        delay(50);
    }
}
int main() {
    int xc, yc, r;
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TurboC3\\BGI");
    printf("Enter the center of the circle:\n");
    printf("Xc = ");
    scanf("%d", &xc);
    printf("Yc = ");
    scanf("%d", &yc);
    printf("Enter the radius: ");
    scanf("%d", &r);
    circleBres(xc, yc, r);
    getch();
    closegraph();
    return 0;
}
```

Output-13:

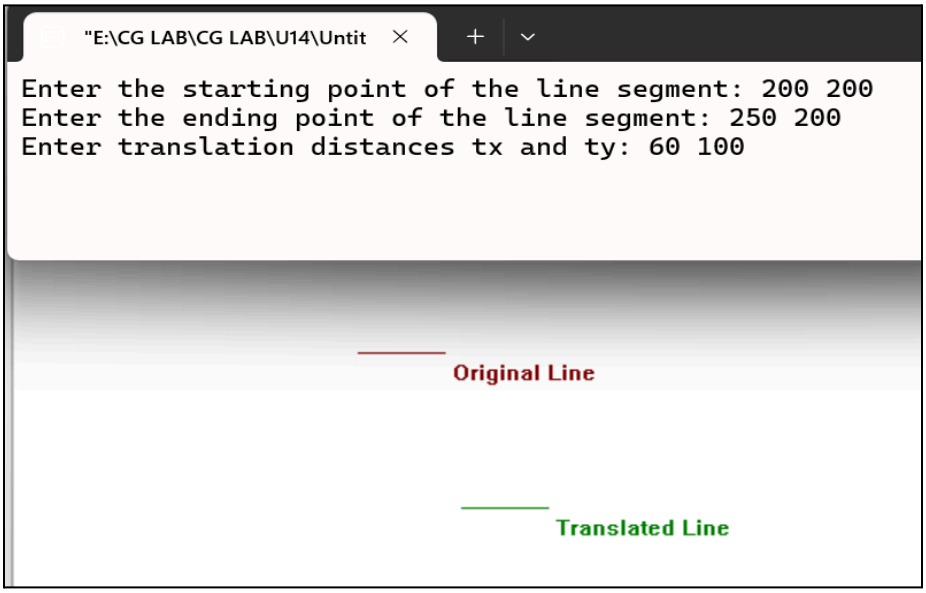


Program – 14: Program for a Lines.

i. Translation of Line

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
int main() {
    int gd = DETECT, gm;
    int x1, y1, x2, y2;       // Original line points
    int tx, ty;               // Translation distances
    int x3, y3, x4, y4;       // Translated line points
    initgraph(&gd, &gm, "C:\\TurboC3\\BGI");
    printf("Enter the starting point of the line segment: ");
    scanf("%d %d", &x1, &y1);
    printf("Enter the ending point of the line segment: ");
    scanf("%d %d", &x2, &y2);
    printf("Enter translation distances tx and ty: ");
    scanf("%d %d", &tx, &ty);
    // Original line
    setbkcolor(WHITE);
    cleardevice();
    setcolor(RED);
    line(x1, y1, x2, y2);
    outtextxy(x2 + 5, y2 + 5, "Original Line");
    // Translated line
    x3 = x1 + tx;
    y3 = y1 + ty;
    x4 = x2 + tx;
    y4 = y2 + ty;
    setcolor(GREEN);
    line(x3, y3, x4, y4);
    outtextxy(x4 + 5, y4 + 5, "Translated Line");
    getch();
    closegraph();
    return 0;
}
```

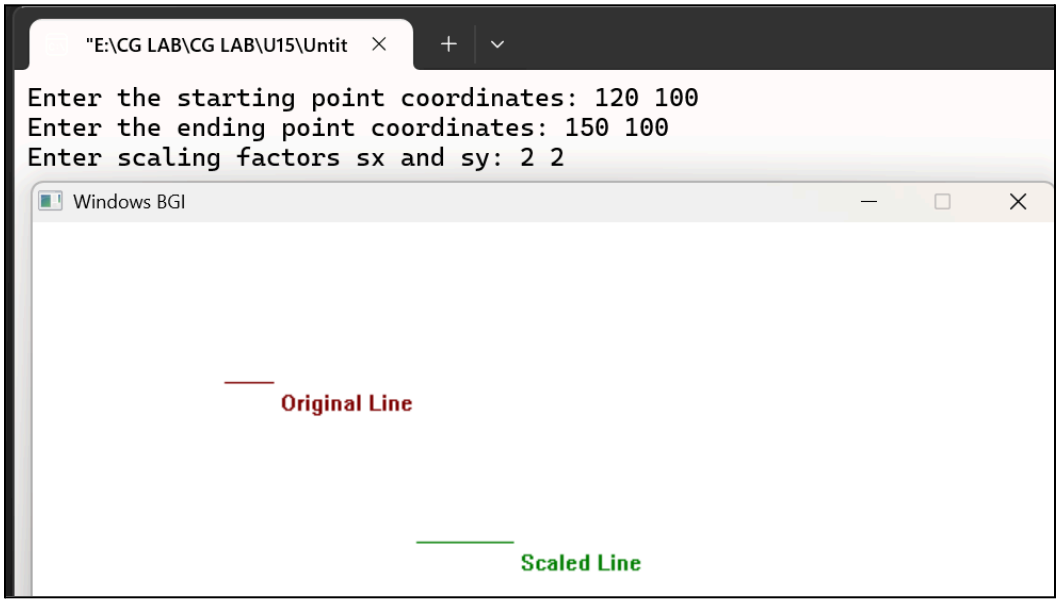
Output-14(i):



14-ii. Scaling of Line.

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
int main() {
    int gd = DETECT, gm;
    float x1, y1, x2, y2;       // Original coordinates
    float sx, sy;               // Scaling factors
    float x3, y3, x4, y4;       // Scaled coordinates
    initgraph(&gd, &gm, "C:\\TurboC3\\BGI");
    printf("Enter the starting point coordinates: ");
    scanf("%f %f", &x1, &y1);
    printf("Enter the ending point coordinates: ");
    scanf("%f %f", &x2, &y2);
    printf("Enter scaling factors sx and sy: ");
    scanf("%f %f", &sx, &sy);
    // Original line
    setbkcolor(WHITE);
    cleardevice();
    setcolor(RED);
    line(x1, y1, x2, y2);
    outtextxy(x2 + 5, y2 + 5, "Original Line");
    // Apply scaling
    x3 = x1 * sx;
    y3 = y1 * sy;
    x4 = x2 * sx;
    y4 = y2 * sy;
    // Scaled line
    setcolor(GREEN);
    line(x3, y3, x4, y4);
    outtextxy(x4 + 5, y4 + 5, "Scaled Line");
    getch();
    closegraph();
    return 0;
}
```

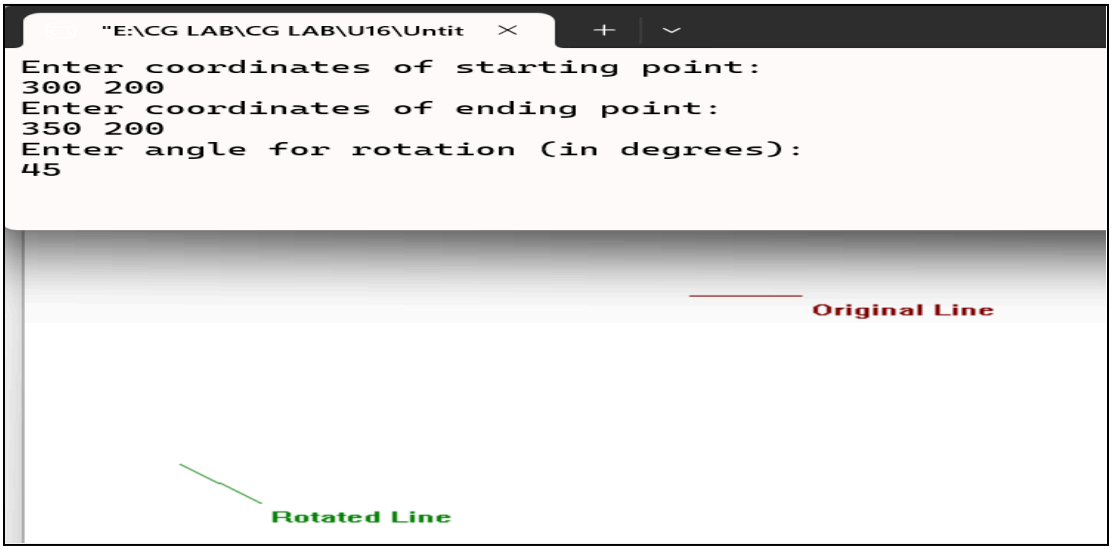
Output-14(ii):



14-iii. Rotation of Line.

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <math.h>
int main() {
    int gd = DETECT, gm;
    float x1, y1, x2, y2;       // Original coordinates
    float x3, y3, x4, y4;       // Rotated coordinates
    float angle, rad;
    initgraph(&gd, &gm, "C:\\TurboC3\\BGI");
    printf("Enter coordinates of starting point:\n");
    scanf("%f %f", &x1, &y1);
    printf("Enter coordinates of ending point:\n");
    scanf("%f %f", &x2, &y2);
    printf("Enter angle for rotation (in degrees):\n");
    scanf("%f", &angle);
    // Original line
    setbkcolor(WHITE);
    cleardevice();
    setcolor(RED);
    line(x1, y1, x2, y2);
    outtextxy(x2 + 5, y2 + 5, "Original Line");
    // Convert angle to radians
    rad = angle * (3.14159 / 180);
    // Rotate both points around origin (0,0)
    x3 = x1 * cos(rad) - y1 * sin(rad);
    y3 = x1 * sin(rad) + y1 * cos(rad);
    x4 = x2 * cos(rad) - y2 * sin(rad);
    y4 = x2 * sin(rad) + y2 * cos(rad);
    // Rotated line
    setcolor(GREEN);
    line(x3, y3, x4, y4);
    outtextxy(x4 + 5, y4 + 5, "Rotated Line");
    getch();
    closegraph();
    return 0;
}
```

Output-14(iii):

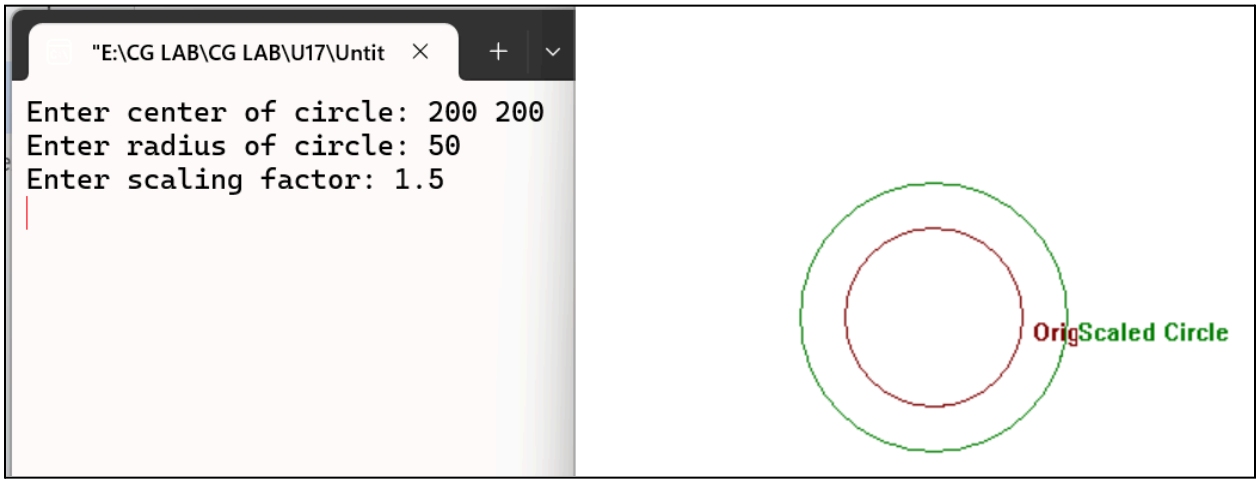


15. Write a C++ program to perform 2D Scaling of a:

15-i: Circle

```
#include <graphics.h>
#include <conio.h>
#include <iostream>
using namespace std;
int main() {
    int gd = DETECT, gm;
    int xc, yc, r;
    float scale;
    initgraph(&gd, &gm, "C:\\TurboC3\\BGI");
    cout << "Enter center of circle: ";
    cin >> xc >> yc;
    cout << "Enter radius of circle: ";
    cin >> r;
    cout << "Enter scaling factor: ";
    cin >> scale;
    setbkcolor(WHITE);
    cleardevice();
    setcolor(RED);
    circle(xc, yc, r);
    outtextxy(xc + r + 5, yc, "Original Circle");
    setcolor(GREEN);
    circle(xc, yc, r * scale);
    outtextxy(xc + r * scale + 5, yc, "Scaled Circle");
    getch();
    closegraph();
    return 0;
}
```

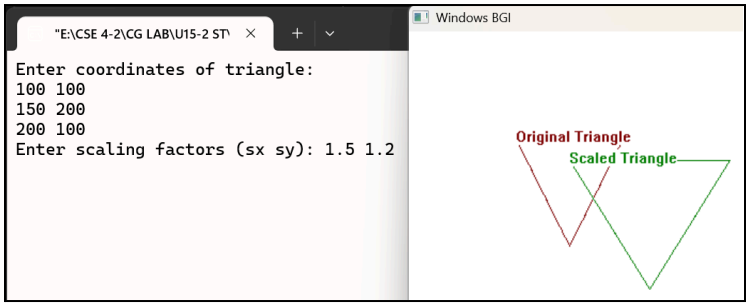
Output-15(i):



15-ii: Triangle.

```
#include <graphics.h>
#include <conio.h>
#include <iostream>
using namespace std;
int main() {
    int gd = DETECT, gm;
    float x1, y1, x2, y2, x3, y3, sx, sy;
    initgraph(&gd, &gm,
"C:\\TurboC3\\BGI");
    cout << "Enter coordinates of
triangle:\n";
    cin >> x1 >> y1 >> x2 >> y2 >> x3 >>
y3;
    cout << "Enter scaling factors (sx sy):
";
    cin >> sx >> sy;
    setbkcolor(WHITE);
    cleardevice();
    setcolor(RED);
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
    outtextxy(x1, y1 - 10, "Original
Triangle");
    x1 *= sx; y1 *= sy;
    x2 *= sx; y2 *= sy;
    x3 *= sx; y3 *= sy;
    setcolor(GREEN);
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
    outtextxy(x1, y1 - 10, "Scaled
Triangle");
    getch();
    closegraph();
    return 0;
}
```

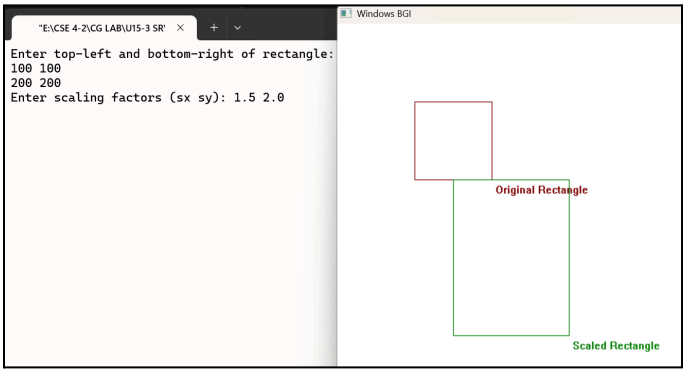
Output-15(ii):



15-iii: Rectangle.

```
#include <graphics.h>
#include <conio.h>
#include <iostream>
using namespace std;
int main() {
    int gd = DETECT, gm;
    float x1, y1, x2, y2, sx, sy;
    initgraph(&gd, &gm,
"C:\\TurboC3\\BGI");
    cout << "Enter top-left and
bottom-right of rectangle:\n";
    cin >> x1 >> y1 >> x2 >> y2;
    cout << "Enter scaling factors (sx sy):
";
    cin >> sx >> sy;
    setbkcolor(WHITE);
    cleardevice();
    setcolor(RED);
    rectangle(x1, y1, x2, y2);
    outtextxy(x2 + 5, y2 + 5, "Original
Rectangle");
    x1 *= sx; y1 *= sy;
    x2 *= sx; y2 *= sy;
    setcolor(GREEN);
    rectangle(x1, y1, x2, y2);
    outtextxy(x2 + 5, y2 + 5, "Scaled
Rectangle");
    getch();
    closegraph();
    return 0;
}
```

Output-15(iii)



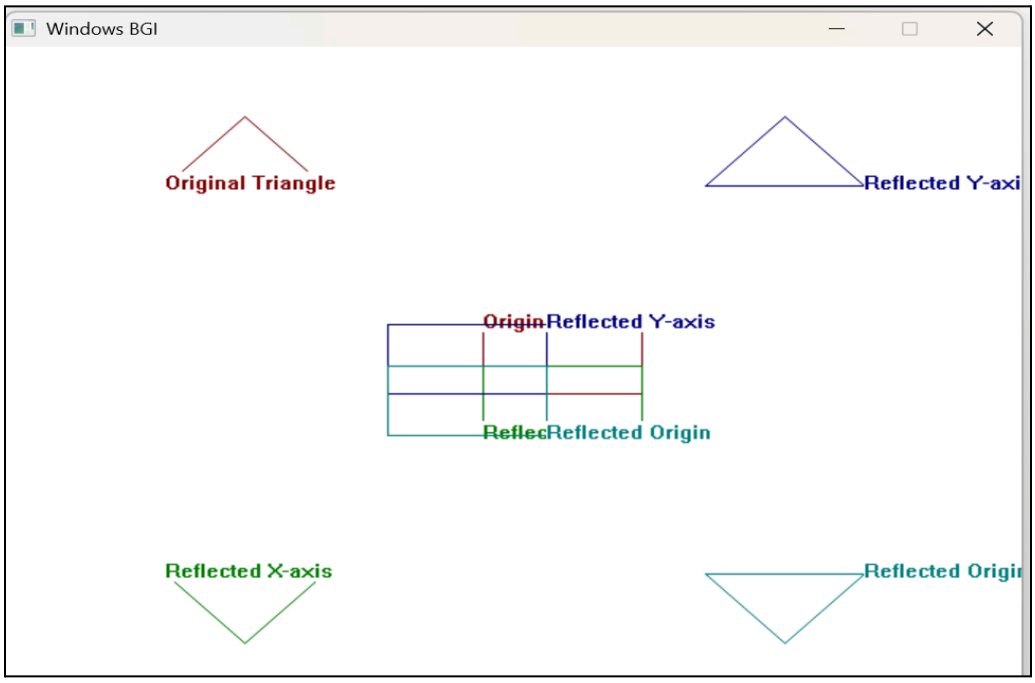
16. Develop a C++ program to perform 2D Reflection of a:

- **i) Circle**
- **ii) Triangle**
- **iii) Rectangle**

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
// Function to reflect a circle over X, Y, and Origin
void reflectCircle(int xc, int yc, int r) {
    // Original Circle
    setcolor(RED);
    circle(xc, yc, r);
    outtextxy(xc + r + 5, yc, "Original Circle");
    // Reflection over X-axis
    setcolor(GREEN);
    circle(xc, -yc + 480, r);
    outtextxy(xc + r + 5, -yc + 480, "Reflected over X-axis");
    // Reflection over Y-axis
    setcolor(BLUE);
    circle(640 - xc, yc, r);
    outtextxy(640 - xc + r + 5, yc, "Reflected over Y-axis");
    // Reflection over Origin
    setcolor(CYAN);
    circle(640 - xc, -yc + 480, r);
    outtextxy(640 - xc + r + 5, -yc + 480, "Reflected over Origin");
}
// Function to reflect a triangle
void reflectTriangle(int x1, int y1, int x2, int y2, int x3, int y3) {
    // Original Triangle
    setbkcolor(WHITE);
    cleardevice();
    setcolor(RED);
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
    outtextxy(x1, y1 - 10, "Original Triangle");
    // X-axis
    setcolor(GREEN);
    line(x1, -y1 + 480, x2, -y2 + 480);
    line(x2, -y2 + 480, x3, -y3 + 480);
    line(x3, -y3 + 480, x1, -y1 + 480);
    outtextxy(x1, -y1 + 480 - 10, "Reflected X-axis");
    // Y-axis
    setcolor(BLUE);
    line(640 - x1, y1, 640 - x2, y2);
    line(640 - x2, y2, 640 - x3, y3);
    line(640 - x3, y3, 640 - x1, y1);
    outtextxy(640 - x1, y1 - 10, "Reflected Y-axis");
    // Origin
    setcolor(CYAN);
    line(640 - x1, -y1 + 480, 640 - x2, -y2 + 480);
    line(640 - x2, -y2 + 480, 640 - x3, -y3 + 480);
    line(640 - x3, -y3 + 480, 640 - x1, -y1 + 480);
    outtextxy(640 - x1, -y1 + 480 - 10, "Reflected Origin");
}
```

```
// Function to reflect a rectangle
void reflectRectangle(int x1, int y1, int x2, int y2) {
    // Original
    setcolor(RED);
    rectangle(x1, y1, x2, y2);
    outtextxy(x1, y1 - 10, "Original Rectangle");
    // X-axis
    setcolor(GREEN);
    rectangle(x1, -y1 + 480, x2, -y2 + 480);
    outtextxy(x1, -y1 + 480 - 10, "Reflected X-axis");
    // Y-axis
    setcolor(BLUE);
    rectangle(640 - x1, y1, 640 - x2, y2);
    outtextxy(640 - x1, y1 - 10, "Reflected Y-axis");
    // Origin
    setcolor(CYAN);
    rectangle(640 - x1, -y1 + 480, 640 - x2, -y2 + 480);
    outtextxy(640 - x1, -y1 + 480 - 10, "Reflected Origin");
}
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TurboC3\\BGI");
    // Reflecting a Circle
    reflectCircle(150, 150, 40);
    // Reflecting a Triangle
    reflectTriangle(100, 100, 150, 50, 200, 100);
    // Reflecting a Rectangle
    reflectRectangle(300, 200, 400, 250);
    getch();
    closegraph();
    return 0;
}
```

Output-16(i,ii,iii) :

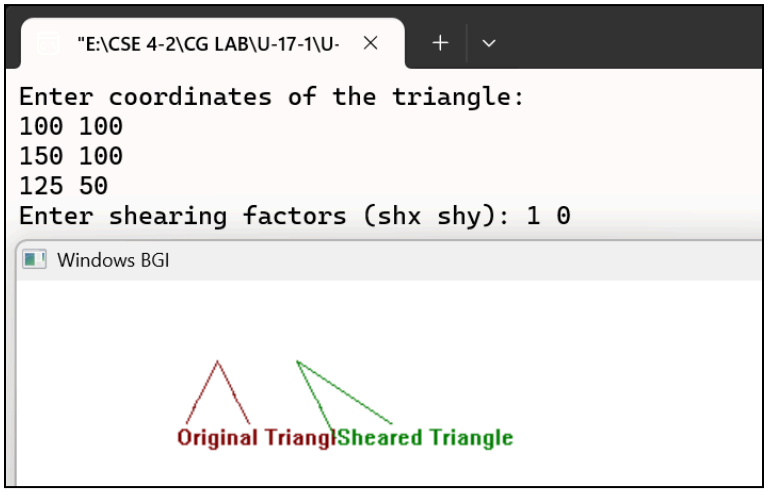


17. Implement a C++ program to perform 2D Shearing of a:

17-i: Triangle

```
#include <graphics.h>
#include <conio.h>
#include <iostream>
using namespace std;
int main() {
    int gd = DETECT, gm;
    float x1, y1, x2, y2, x3, y3;
    float shx, shy;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
    // Input coordinates of the triangle
    cout << "Enter coordinates of the triangle:\n";
    cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
    // Input shearing factors
    cout << "Enter shearing factors (shx shy): ";
    cin >> shx >> shy;
    // Draw original triangle
    setbkcolor(WHITE);
    cleardevice();
    setcolor(RED);
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
    outtextxy(x1, y1 - 10, "Original Triangle");
    // Apply shearing
    float x1_new = x1 + shx * y1;
    float y1_new = y1 + shy * x1;
    float x2_new = x2 + shx * y2;
    float y2_new = y2 + shy * x2;
    float x3_new = x3 + shx * y3;
    float y3_new = y3 + shy * x3;
    // Draw sheared triangle
    setcolor(GREEN);
    line(x1_new, y1_new, x2_new, y2_new);
    line(x2_new, y2_new, x3_new, y3_new);
    line(x3_new, y3_new, x1_new, y1_new);
    outtextxy(x1_new, y1_new - 10, "Sheared Triangle");
    getch();
    closegraph();
    return 0;
}
```

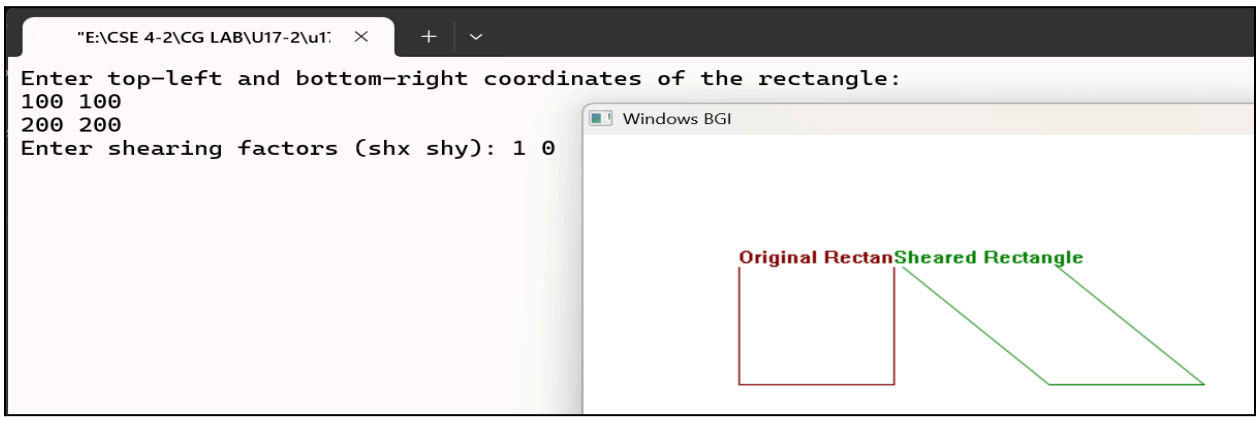
Output-17(i) :



17(ii) Rectangle.

```
#include <graphics.h>
#include <conio.h>
#include <iostream>
using namespace std;
int main() {
    int gd = DETECT, gm;
    float x1, y1, x2, y2; // Opposite corners of the rectangle
    float shx, shy;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
    // Input rectangle coordinates
    cout << "Enter top-left and bottom-right coordinates of the rectangle:\n";
    cin >> x1 >> y1 >> x2 >> y2;
    // Input shearing factors
    cout << "Enter shearing factors (shx shy): ";
    cin >> shx >> shy;
    // Original rectangle coordinates
    float x3 = x2, y3 = y1; // top-right
    float x4 = x1, y4 = y2; // bottom-left
    // Draw original rectangle
    setbkcolor(WHITE);
    cleardevice();
    setcolor(RED);
    rectangle(x1, y1, x2, y2);
    outtextxy(x1, y1 - 10, "Original Rectangle");
    // Apply shearing
    float sx1 = x1 + shx * y1;
    float sy1 = y1 + shy * x1;
    float sx2 = x2 + shx * y2;
    float sy2 = y2 + shy * x2;
    float sx3 = x3 + shx * y3;
    float sy3 = y3 + shy * x3;
    float sx4 = x4 + shx * y4;
    float sy4 = y4 + shy * x4;
    // Draw sheared rectangle (manually as a polygon)
    setcolor(GREEN);
    line(sx1, sy1, sx3, sy3); // Top
    line(sx3, sy3, sx2, sy2); // Right
    line(sx2, sy2, sx4, sy4); // Bottom
    line(sx4, sy4, sx1, sy1); // Left
    outtextxy(sx1, sy1 - 10, "Sheared Rectangle");
    getch();
    closegraph();
    return 0;
}
```

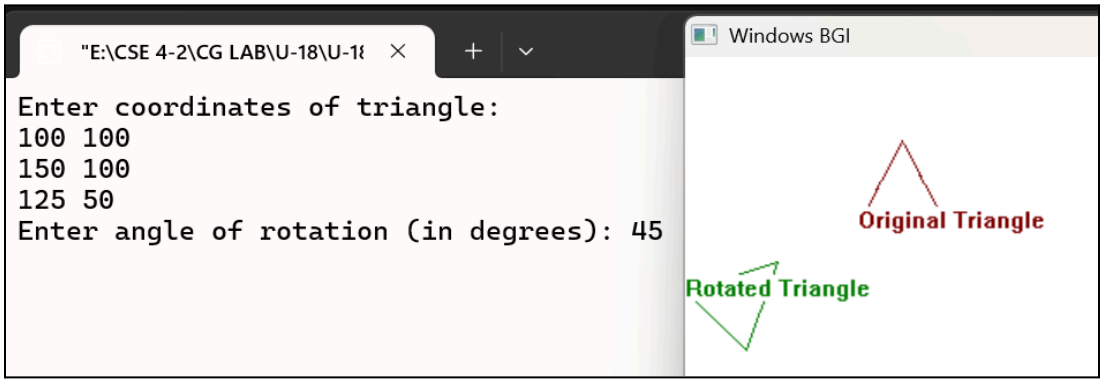
Output-17(ii)



18. Write a C++ program to perform 2D Anticlockwise Rotation of any shape by a given angle.

```
#include <graphics.h>
#include <conio.h>
#include <iostream>
#include <cmath>
using namespace std;
// Function to rotate a point (x, y) around origin by angle in degrees
void rotatePoint(float x, float y, float angle, float &xr, float &yr) {
    float rad = angle * M_PI / 180.0; // Convert to radians
    xr = x * cos(rad) - y * sin(rad);
    yr = x * sin(rad) + y * cos(rad);
}
int main() {
    int gd = DETECT, gm;
    float x1, y1, x2, y2, x3, y3;
    float angle;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
    // Input triangle coordinates
    cout << "Enter coordinates of triangle:\n";
    cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
    // Input rotation angle
    cout << "Enter angle of rotation (in degrees): ";
    cin >> angle;
    // Draw original triangle
    setbkcolor(WHITE);
    cleardevice();
    setcolor(RED);
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
    outtextxy(x1, y1 - 10, "Original Triangle");
    // Rotate each point around origin (0,0)
    float x1r, y1r, x2r, y2r, x3r, y3r;
    rotatePoint(x1, y1, angle, x1r, y1r);
    rotatePoint(x2, y2, angle, x2r, y2r);
    rotatePoint(x3, y3, angle, x3r, y3r);
    // Draw rotated triangle
    setcolor(GREEN);
    line(x1r, y1r, x2r, y2r);
    line(x2r, y2r, x3r, y3r);
    line(x3r, y3r, x1r, y1r);
    outtextxy(x1r, y1r - 10, "Rotated Triangle");
    getch();
    closegraph();
    return 0;
}
```

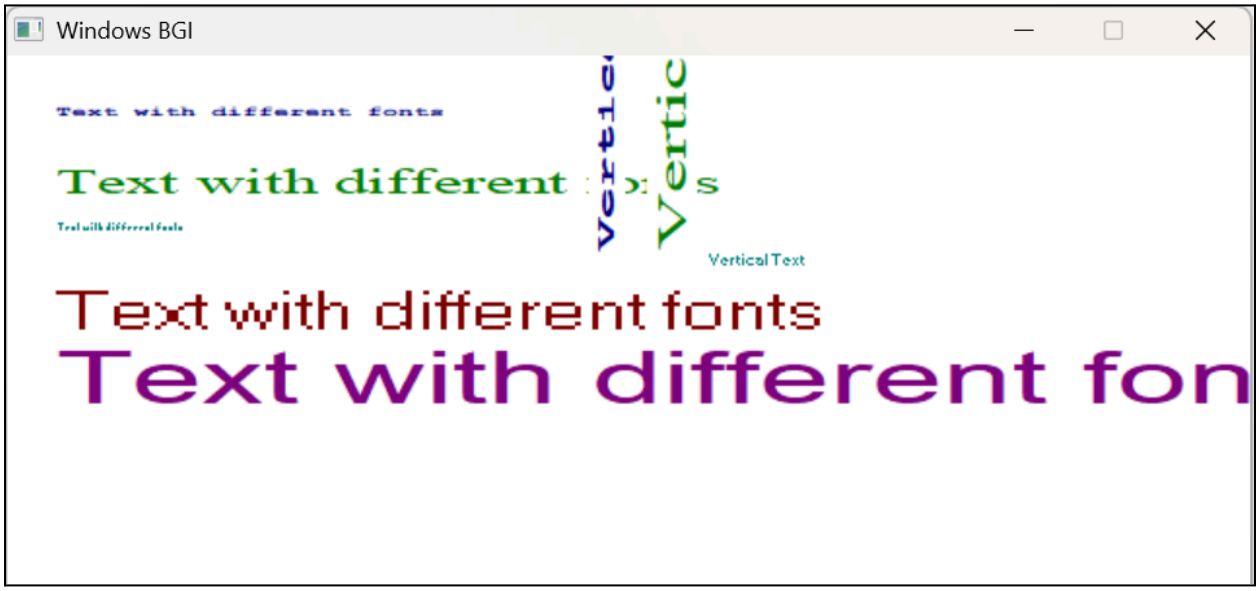
Output-18:



Program – 19: Creating various types of texts and fonts.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
int main()
{
int gd=DETECT,gm,x=25,y=25,font=10;
initgraph(&gd,&gm,"C:\\turboC3\\BGI");
for(font=0;font<=4;font++)
{
settextstyle(font,HORIZ_DIR,font+1);// sets font type, font direction, size
setcolor(font+1); // sets color for text.
outtextxy(x,y,"text with different fonts"); // prints message on screen at (x,y)
y=y+25;
}
for(font=0;font<=2;font++)
{
settextstyle(font,VERT_DIR,font+2);
setcolor(font+1);
x=250;
y=100;
outtextxy(x,y,"text in vertical direction");
y=y+25;
}
getch();
closegraph();
}
```

Output-19:



Program – 20: Program for Window to Viewport Transformation.

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
int main() {
    float xwmin, xwmax, ywmin, ywmax;
    float xvmin, xvmax, yvmin, yvmax;
    float x[10], y[10];
    float sx, sy;
    int gd = DETECT, gm, i;
    printf("Enter window coordinates (xwmin, ywmin, xwmax, ywmax):\n");
    scanf("%f%f%f%f", &xwmin, &ywmin, &xwmax, &ywmax);
    printf("Enter viewport coordinates (xvmin, yvmin, xvmax, yvmax):\n");
    scanf("%f%f%f%f", &xvmin, &yvmin, &xvmax, &yvmax);
    printf("Enter vertices for triangle:\n");
    for (i = 0; i < 3; i++) {
        printf("Enter (x%d, y%d): ", i, i);
        scanf("%f%f", &x[i], &y[i]);
    }
    // Calculate scaling factors
    sx = (xvmax - xvmin) / (xwmax - xwmin);
    sy = (yvmax - yvmin) / (ywmax - ywmin);  // <- Fixed typo
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
     setbkcolor(BLUE);
    cleardevice();
    // Draw window and triangle inside it
    outtextxy(80, 30, "Window Port");
    rectangle(xwmin, ywmin, xwmax, ywmax);\
    for (i = 0; i < 2; i++) {
        line(x[i], y[i], x[i + 1], y[i + 1]);
    }
    line(x[2], y[2], x[0], y[0]);
    getch();
    cleardevice();
    // Perform transformation to viewport
    for (i = 0; i < 3; i++) {
        x[i] = xvmin + (x[i] - xwmin) * sx;
        y[i] = yvmin + (y[i] - ywmin) * sy;
    }
    // Draw viewport and transformed triangle
    outtextxy(150, 10, "View Port");
    rectangle(xvmin, yvmin, xvmax, yvmax);
    for (i = 0; i < 2; i++) {
        line(x[i], y[i], x[i + 1], y[i + 1]);
    }
    line(x[2], y[2], x[0], y[0]);
    getch();
    closegraph();
    return 0;
}
```

Output-20: