Experiment-7

Student Name: Mayank Singh **UID: 22BCS10205**
Branch: CSE **Section/Group: 22BCS_IOT-612/B**
Semester: 6 **Date of Performance: 13-03-25**
Subject Name: Computer Graphics Lab **Subject Code: 22CSH-352**

1. **Aim:**

Evaluate the 4-bit region code for line endpoints and determine whether the line lies inside or outside the screen.

2. **Objective:**

To calculate and display the 4-bit region code for line endpoints and determine whether the line lies within the screen boundaries.

3. **Algorithm:**

Step 1: Initialize Graphics Mode

- Detect and initialize the graphics mode using initgraph().
- Define screen boundaries (xmin, ymin, xmax, ymax).
- Draw boundary lines on the screen to represent the clipping window.

Step 2: Input Line Endpoints

- Accept user input for the coordinates of two endpoints (x0, y0 and x1, y1).
- Draw the line on the screen.

Step 3: Compute Region Codes

- For each endpoint (x, y), compute the 4-bit region code:
  - bit1 (Top - 8) → 1 if y < ymin, else 0.
  - bit2 (Bottom - 4) → 1 if y > ymax, else 0.
  - bit3 (Right - 2) → 1 if x > xmax, else 0.
  - bit4 (Left - 1) → 1 if x < xmin, else 0.
- Display the computed region code for each endpoint.

Step 4: Determine Line Position

- Completely Inside (Trivial Accept) → If both endpoints have region code 0000.
- Completely Outside (Trivial Reject) → If AND of both region codes is nonzero (num[0] & num[1] ≠ 0).
- Partially Inside (Clipping Required) → If OR of both region codes is nonzero (num[0] | num[1] ≠ 0) and AND is zero.

Step 5: Display Result

- Print whether the line is fully inside, requires clipping, or is completely outside the window.
- Wait for user input (getch()) before closing the program.

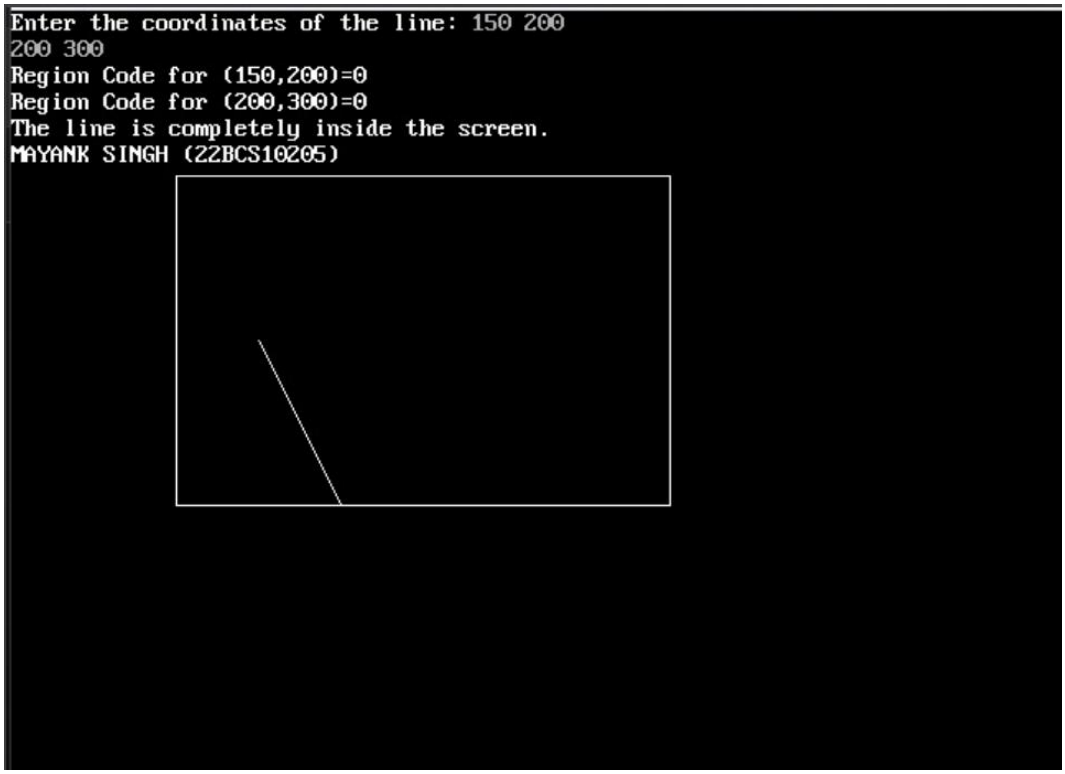
4. Implementation/Code:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
const int xmin = 100, ymin = 100, xmax = 400, ymax = 300;

int getRegionCode(int x, int y) {
   int code = 0;
   if (y > ymax) code |= 8;
   if (y < ymin) code |= 4;
   if (x > xmax) code |= 2;
   if (x < xmin) code |= 1;
   return code;
}

void main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
   int x1, y1, x2, y2;
   cout << "Enter the coordinates of the line: ";
   cin >> x1 >> y1 >> x2 >> y2;
   int code1 = getRegionCode(x1, y1);
   int code2 = getRegionCode(x2, y2);
   cout << "Region Code for (" << x1 << "," << y1 << ")=" << code1<<"\n";
   cout << "Region Code for (" << x2 << "," << y2 << ")=" << code2<<"\n";
   setcolor(WHITE);
   rectangle(xmin, ymin, xmax, ymax);
   setcolor(WHITE);
   line(x1, y1, x2, y2);
   if (code1 == 0 && code2 == 0)
      cout << "The line is completely inside the screen.\n";
   else if ((code1 & code2) != 0)
      cout << "The line is completely outside the screen.\n";
   else
      cout << "The line is partially inside and needs clipping.\n";
   cout<<"MAYANK SINGH  (22BCS10205)";
   getch();
   closegraph();
}
```

5. Output:-



6. Learning Outcomes:-

- Learnt how to compute 4-bit region codes for line endpoints based on their position relative to the clipping window.
- Understood how to determine whether a line is fully inside, partially inside, or outside a given window using bitwise operations.
- Gained practical knowledge of bitwise AND and OR operations to classify lines based on region codes.
- Learnt how to use graphics.h to visualize screen boundaries, lines, and region codes in Turbo C++.
- Understood the significance of clipping algorithms in rendering, game development, and graphical user interfaces.