Name: Khushi Nitinkumar Patel
PRN: 2020BTECS00037
Batch: T5

Experiment 10: Implementation of all Line Clipping algorithms

- # Cohen-Sutherland algorithm

Code

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <graphics.h>
#include <dos.h>

typedef struct coord
{
   int x, y;
   char code[4];
} PT;

void drawwindow();
void drawline(PT p1, PT p2);
PT setcode(PT p);
int visibility(PT p1, PT p2);
PT resetendpt(PT p1, PT p2);

int main()
{
   int gd = DETECT, v, gm;
   PT p1, p2, p3, p4, ptemp;

   printf("\nEnter x1 and y1\n");
   scanf("%d %d", &p1.x, &p1.y);
   printf("\nEnter x2 and y2\n");
   scanf("%d %d", &p2.x, &p2.y);

   initgraph(&gd, &gm, "c:\\turboc3\\bgi");
```

```
    drawwindow();
    delay(1000);

    drawline(p1, p2);
    delay(5000);
    cleardevice();

    delay(1000);
    p1 = setcode(p1);
    p2 = setcode(p2);
    v = visibility(p1, p2);
    delay(1000);

    switch (v)
    {
    case 0:
       drawwindow();
       delay(500);
       drawline(p1, p2);
       break;
    case 1:
       drawwindow();
       delay(500);
       break;
    case 2:
       p3 = resetendpt(p1, p2);
       p4 = resetendpt(p2, p1);
       drawwindow();
       delay(500);
       drawline(p3,  p4);
       break;
    }

    delay(5000);
    closegraph();
}

void drawwindow()
{
    line(150, 100, 450, 100);
    line(450, 100, 450, 350);
    line(450, 350, 150, 350);
    line(150, 350, 150, 100);
}

void drawline(PT p1, PT p2)
{
```

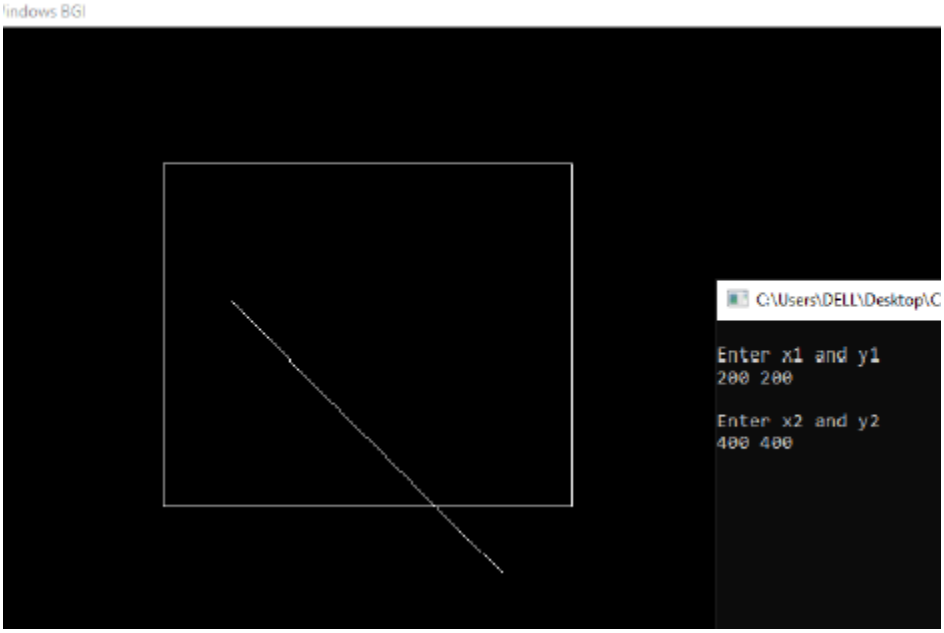
```
    line(p1.x, p1.y, p2.x, p2.y);
}

PT setcode(PT p)
{
    PT ptemp;

    if (p.y < 100)
        ptemp.code[0] = '1'; // Top
    else
        ptemp.code[0] = '0';

    if (p.y > 350)
        ptemp.code[1] = '1'; // Bottom
    else
        ptemp.code[1] = '0';

    if (p.x > 450)
        ptemp.code[2] = '1'; // Right
    else
        ptemp.code[2] = '0';

    if (p.x < 150)
        ptemp.code[3] = '1'; // Left
    else
        ptemp.code[3] = '0';

    ptemp.x = p.x;
    ptemp.y = p.y;

    return (ptemp);
}

int visibility(PT p1, PT p2)
{
    int i, flag = 0;

    for (i = 0; i < 4; i++)
    {
        if ((p1.code[i] != '0') || (p2.code[i] != '0'))
            flag = 1;
    }

    if (flag == 0)
        return (0);
```

```
    for (i = 0; i < 4; i++)
    {
        if ((p1.code[i] == p2.code[i]) && (p1.code[i] == '1'))
            flag = '0';
    }

    if (flag == 0)
        return (1);

    return (2);
}

PT resetendpt(PT p1, PT p2)
{
    PT   temp;
    int  x, y, i;
    float m, k;

    if (p1.code[3] == '1')
        x = 150;

    if (p1.code[2] == '1')
        x = 450;

    if ((p1.code[3] == '1') || (p1.code[2] == '1'))
    {
        m = (float)(p2.y - p1.y) / (p2.x - p1.x);
        k = (p1.y + (m * (x - p1.x)));
        temp.y = k;
        temp.x = x;

        for (i = 0; i < 4; i++)
            temp.code[i] = p1.code[i];

        if (temp.y <= 350 && temp.y >= 100)
            return (temp);
    }

    if (p1.code[0] == '1')
        y = 100;

    if (p1.code[1] == '1')
        y = 350;
```

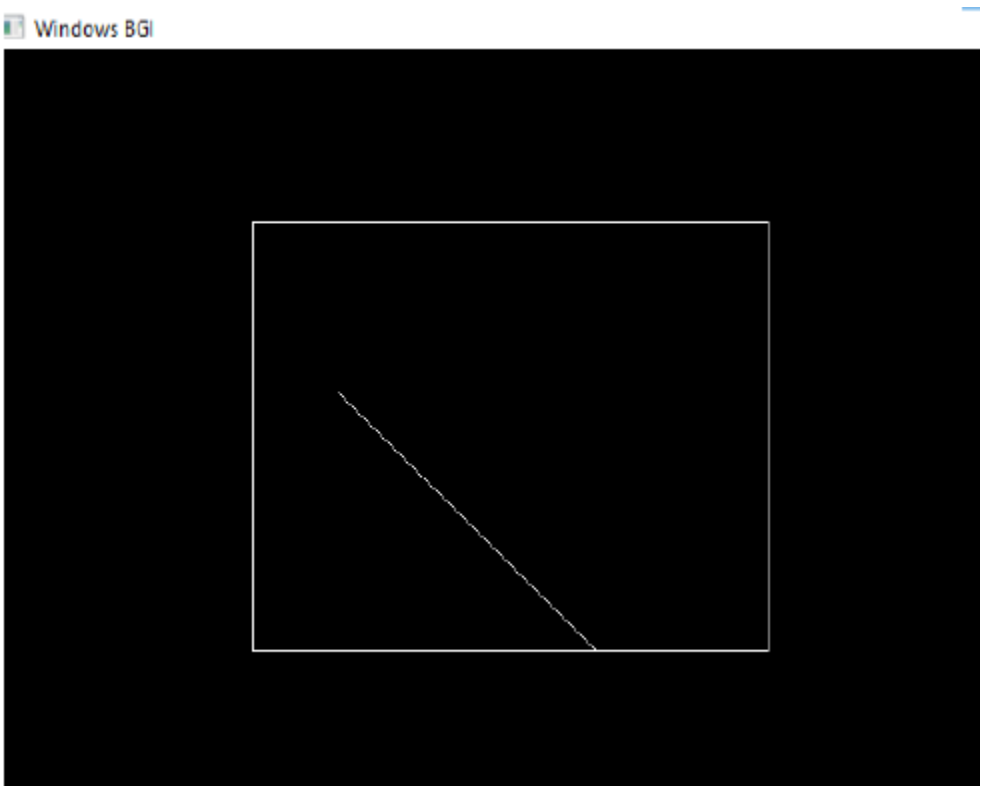
```
    if ((p1.code[0] == '1') || (p1.code[1] == '1'))
    {
        m = (float)(p2.y - p1.y) / (p2.x - p1.x);
        k = (float)p1.x + (float)(y - p1.y) / m;
        temp.x = k;
        temp.y = y;

        for (i = 0; i < 4; i++)
            temp.code[i] = p1.code[i];

        return (temp);
    }
    else
        return (p1);
}
```

Output

Before Clipping



After Clipping



- **Midpoint Subdivision Algorithm:**

Code

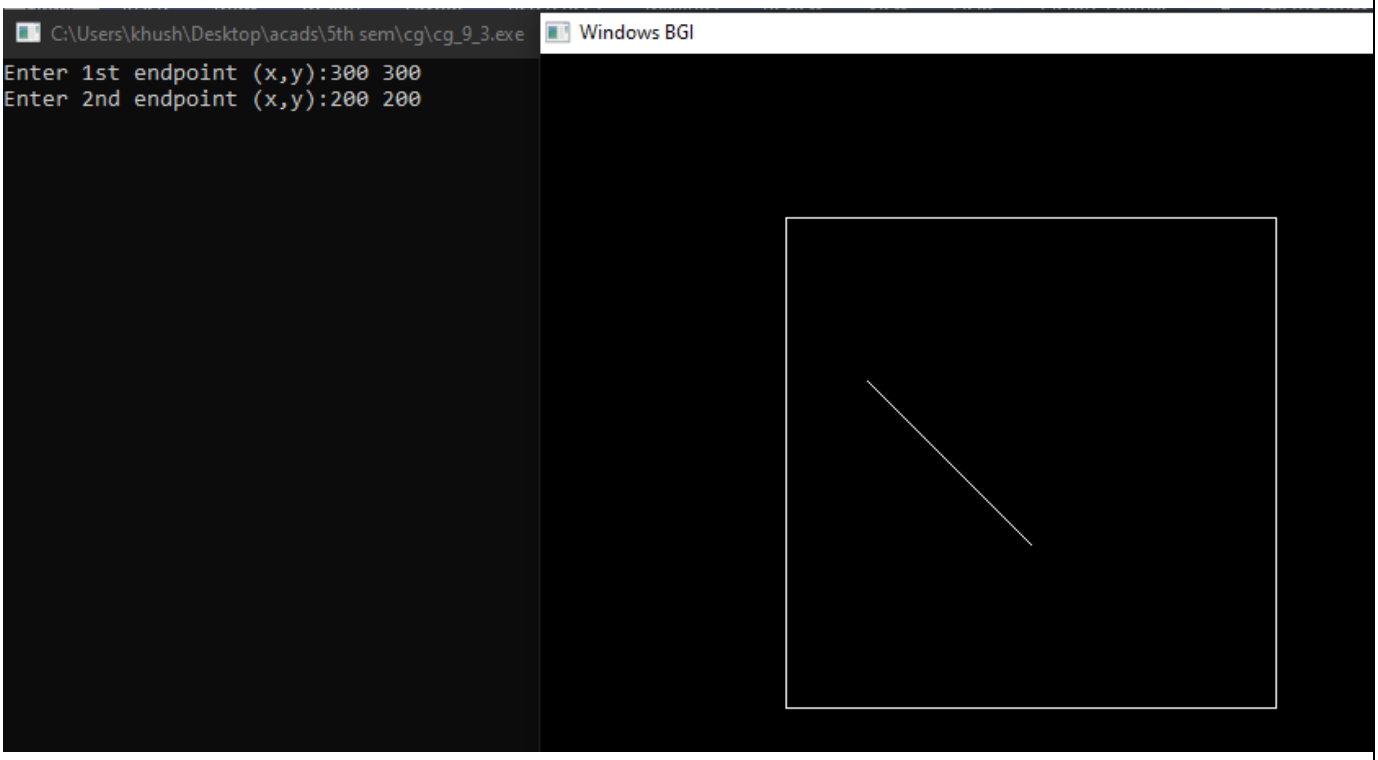
```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <dos.h>
#include <math.h>
#include <graphics.h>
typedef struct coordinate
{
   int x, y;
   char code[4];
} PT;
void drawwindow();
void drawline(PT p1, PT p2);
PT setcode(PT p);
int visibility(PT p1, PT p2);
PT resetendpt(PT p1, PT p2);
void drawwindow()
{
   setcolor(WHITE);

   line(150, 100, 450, 100);
   line(450, 100, 450, 400);
   line(450, 400, 150, 400);
   line(150, 400, 150, 100);
}
```

```
void drawline(PT p1, PT p2)
{
   setcolor(15);
   line(p1.x, p1.y, p2.x, p2.y);
}
int visibility(PT p1, PT p2)
{
   int i, flag = 0;
   for (i = 0; i < 4; i++)
   {
      if ((p1.code[i] != '0') || (p2.code[i] != '0'))
         flag = 1;
   }
   if (flag == 0)
      return (0);
   for (i = 0; i < 4; i++)
   {
      if ((p1.code[i] == p2.code[i]) && (p1.code[i] == '1'))
         flag = 0;
   }
   if (flag == 0)
      return (1);
   else
   {
      return 2;
   }
}
PT setcode(PT p)
{
   PT ptemp;
   if (p.y <= 100)
      ptemp.code[0] = '1';
   else
      ptemp.code[0] = '0';
   if (p.y >= 400)
      ptemp.code[1] = '1';
   else
      ptemp.code[1] = '0';
   if (p.x >= 450)
      ptemp.code[2] = '1';
   else
      ptemp.code[2] = '0';
```

```
    if (p.x <= 150)
        ptemp.code[3] = '1';
    else
        ptemp.code[3] = '0';
    ptemp.x = p.x;
    ptemp.y = p.y;
    return (ptemp);
}
midsub(PT p1, PT p2)
{
    PT mid;
    int v;
    setcolor(YELLOW);
    p1 = setcode(p1);
    p2 = setcode(p2);
    v = visibility(p1, p2);
    switch (v)
    {
    case 0:
        drawline(p1, p2);
        break;
    case 1:
        break;
    case 2:
        mid.x = p1.x + (p2.x - p1.x) / 2;
        mid.y = p1.y + (p2.y - p1.y) / 2;
        midsub(p1, mid);
        mid.x = mid.x + 1;
        mid.y = mid.y + 1;
        midsub(mid, p2);
        break;
    }
}
int main()
{
    int gd = DETECT, gm, v;
    PT p1, p2, ptemp;
    initgraph(&gd, &gm, "c:\\turboc3\\bgi");
    cleardevice();
    printf("Enter 1st endpoint (x,y):");
    scanf("%d %d", &p1.x, &p1.y);
    printf("Enter 2nd endpoint (x,y):");
    scanf("%d %d", &p2.x, &p2.y);
    drawwindow();
    midsub(p1, p2);
    getch();
    closegraph();
```

```
    return (0);
}
```

Output



- ## Lian Barsky

Code

```
// Lian Barsky Algo
#include <stdio.h>
#include <graphics.h>
#include <math.h>
#include <dos.h>
int main()
{
   int i, gd = DETECT, gm;
   int x1, y1, x2, y2, xmin, xmax, ymin, ymax, xx1, xx2, yy1, yy2, dx, dy;
   float t1, t2, p[4], q[4], temp;
   x1 = 150;
   y1 = 100;
   x2 = 250;
   y2 = 200;
   xmin = 100;
   ymin = 100;
   xmax = 200;
   ymax = 200;
```

```
initgraph(&gd, &gm, "c:\\turboc3\\bgi");
rectangle(xmin, ymin, xmax, ymax);
dx = x2 - x1;

dy = y2 - y1;
p[0] = -dx;
p[1] = dx;
p[2] = -dy;
p[3] = dy;
q[0] = x1 - xmin;
q[1] = xmax - x1;
q[2] = y1 - ymin;
q[3] = ymax - y1;
for (i = 0; i < 4; i++)
{
   if (p[i] == 0)
   {
      printf("line is parallel to one of the clipping boundary");
      if (q[i] >= 0)
      {
         if (i < 2)
         {
            if (y1 < ymin)
            {
               y1 = ymin;
            }
            if (y2 > ymax)
            {
               y2 = ymax;
            }
            line(x1, y1, x2, y2);
         }
         if (i > 1)
         {
            if (x1 < xmin)
            {
               x1 = xmin;
            }
            if (x2 > xmax)
            {
               x2 = xmax;
            }
```

```
                line(x1, y1, x2, y2);
            }
        }
    }
}
t1 = 0;
t2 = 1;
for (i = 0; i < 4; i++)
{

    temp = q[i] / p[i];
    if (p[i] < 0)
    {
        if (t1 <= temp)
            t1 = temp;
    }
    else
    {
        if (t2 > temp)
            t2 = temp;
    }
}
if (t1 < t2)
{
    xx1 = x1 + t1 * p[1];
    xx2 = x1 + t2 * p[1];
    yy1 = y1 + t1 * p[3];
    yy2 = y1 + t2 * p[3];
    line(xx1, yy1, xx2, yy2);
}
delay(10000);
closegraph();
return 0;
}
```

Output:

- **Cyres BeckCode:**

```
// Cyrus Beck algo
#include <conio.h>
#include <iostream>
#include <graphics.h>
#include <process.h>

#define ROUND(a) ((int)(a + 0.5))
using namespace std;
struct _line
{
   int x1, y1;
   int x2, y2;
};

int xmax, xmin, ymax, ymin;

void clip(_line a)
{
   int p[4], q[4], i, dx, dy, flag = 1;
   double u1 = 0, u2 = 1, temp;
   dx = a.x2 - a.x1;
   dy = a.y2 - a.y1;
   p[0] = -dx;
   q[0] = a.x1 - xmin;
   p[1] = dx;
   q[1] = xmax - a.x1;
   p[2] = -dy;
   q[2] = a.y1 - ymin;
   p[3] = dy;
   q[3] = ymax - a.y1;

   if (p[0] == 0 && p[3] == 0)
   {
      if (a.x1 >= xmin && a.x1 <= xmax && a.y1 >= ymin && a.y1 <= ymax)
         putpixel(a.x1, a.y1, GREEN);
      else
         return;
   }
```

```
    if (p[0] == 0)
        if (q[0] * q[1] <= 0)
            return;
    if (p[2] == 0)
        if (q[2] * q[3] <= 0)
            return;

    for (i = 0; i < 4; i++)
    {
        if (p[i] < 0 && flag)
        {
            temp = (double)q[i] / (double)p[i];
            if (temp > u2)
                flag = 0;
            else if (temp > u1)
                u1 = temp;
        }
        else if (p[i] > 0 && flag)
        {
            temp = (double)q[i] / (double)p[i];
            if (temp < u1)
                flag = 0;
            else if (temp < u2)
                u2 = temp;
        }
    }
    if (u1 >= u2 || flag == 0)
        return;
    temp = a.x1;
    i = a.y1;
    a.x1 = temp + u1 * dx;
    a.x2 = temp + u2 * dx;
    a.y1 = i + u1 * dy;
    a.y2 = i + u2 * dy;
    line(319 + ROUND(a.x1), 240 - ROUND(a.y1), 319 + ROUND(a.x2), 240 -
ROUND(a.y2));
}

void drawWindow(int xmin, int ymin, int xmax, int ymax)
{
    line(319 + xmin, 240 - ymax, 319 + xmax, 240 - ymax); // Top Edge
    line(319 + xmax, 240 - ymax, 319 + xmax, 240 - ymin); // Right Edge
    line(319 + xmax, 240 - ymin, 319 + xmin, 240 - ymin); // Bottom Edge
    line(319 + xmin, 240 - ymin, 319 + xmin, 240 - ymax); // Left Edge
}
```

```
int main()
{
   int gd = DETECT, gm, n, i;
   _line *a;
   initgraph(&gd, &gm, "c:\\turboc3\\bgi");
   cout << "Enter the window coordinates : \n";
   cout << "Lower Left Corner : ";
   cin >> xmin >> ymin;

   cout << "Upper Right Corner : ";
   cin >> xmax >> ymax;
   if (xmax < xmin || ymax < ymin)
   {
      cout << "\nIncorrect Window";
      getch();
      exit(0);
   }

   cout << "How many lines do you want to draw : ";
   cin >> n;
   a = new _line[n];
   cout << "Enter Coordinates : \n";
   for (i = 0; i < n; i++)
   {
      cout << "line " << i + 1 << " : ";
      cout << "Enter coordinates of Ist Vertex" << endl;
      cout << "x1 =";
      cin >> a[i].x1;
      cout << "y1 =";
      cin >> a[i].y1;

      cout << "Enter coordinates of IInd Vertex" << endl;
      cout << "x2 =";
      cin >> a[i].x2;
      cout << "y2 =";
      cin >> a[i].y2;
   }

   initgraph(&gd, &gm, (char *)"");
   outtextxy(0, 5, "The original Line is");

   for (i = 0; i < n; i++)
      line(319 + a[i].x1, 240 - a[i].y1, 319 + a[i].x2, 240 - a[i].y2);
   getch();
```

```
    setcolor(LIGHTGREEN);
    outtextxy(0, 20, "The Clipping Window is");

    drawWindow(xmin, ymin, xmax, ymax);
    getch();
    cleardevice();
    drawWindow(xmin, ymin, xmax, ymax);
    setcolor(WHITE);
    outtextxy(0, 5, "The Clipped line is");
    for (i = 0; i < n; i++)
        clip(a[i]);
    getch();
    closegraph();
    restorecrtmode();
}
```

Output