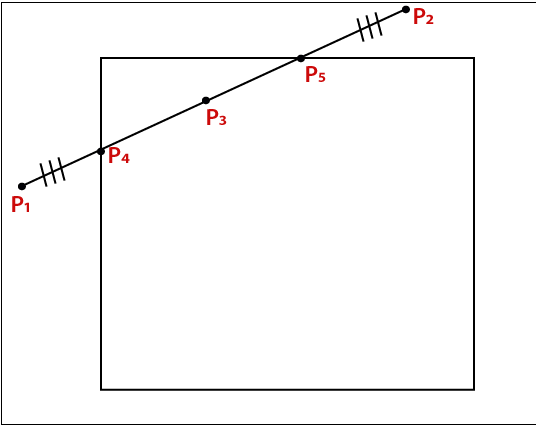
**Practical No 8.B**

**Implementation of MIDPOINT SUBDIVISION LINE clipping algorithms.**

**Aim: Write a program to implement a Midpoint Subdivision line clipping algorithm.**

**Theory:**

Midpoint subdivision algorithm uses the line end point codes and associated tests to immediately identify totally visible lines and trivially invisible lines. In order to fully use this simple algorithm to reject most invisible lines, two sets of inclined parallel orthogonal lines passing through four corners of the original window are constructed, forming a filtering window to carry out the midpoint subdivision algorithm. Repeatedly tilting the filtering window, i.e., changing the inclined angle of its bounding edges in certain way, trivially invisible lines are rejected successively. Theoretical analysis and running test of the new algorithm verified its high efficiency. Parallel architecture could further raise the speed of clipping greatly. Midpoint subdivision algorithm is an extension of the Cyrus Beck algorithm. This algorithm is mainly used to compute visible areas of lines that are present in the view port are of the sector or the image. It follows the principle of the bisection method and works similarly to the Cyrus Beck algorithm by bisecting the line in to equal halves. But unlike the Cyrus Beck algorithm, which only bisects the line once, Midpoint Subdivision Algorithm bisects the line numerous times. We can calculate the midpoint of the line by the following formula: pm = (p1 + p2)/2



**Algorithm Midpoint Subdivision Line Clipping**

Step 1: Assign the bit code for both endpoints of the line.

Step 2: Now, implement OR operation on both endpoints of the line.

Step 3: If the OR = 0000,

Then

{The line is Visible}

Else

{Implement AND operation on endpoints}

Then

If AND? 0000

Then

{The line is Invisible}

Else

AND = 0000

{The line is the partially visible}

Step 4: For partially visible line, we need to find the midpoint.

Xm = (x1 + x2)/2 (For x coordinate)

Ym = (y1 + y2)/2 (For y coordinate)

Step 5: We need to check that the line is near to the boundary of the window or not.

Step 6: If the line is visible or invisible, then repeat steps 1 to 5.

Step 7: Stop.

**Conclusion: We have implemented Midpoint Subdivision line clipping algorithm.**

**Code:**

#include<iostream.h>

#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 midsub(PT p1,PT p2);

int main()

{

int gd=DETECT, gm,v;

PT p1,p2,ptemp;

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

cleardevice();

cout<<"ENTER END-POINT 1 (x,y):";

cin>>p1.x>>p1.y;

cout<<"ENTER END-POINT 2 (x,y):";

cin>>p2.x>>p2.y;

cleardevice();

drawwindow();

getch();

drawline(p1,p2);

getch();

cleardevice();

drawwindow();

midsub(p1,p2);

getch();

closegraph();

return(0);

}

void midsub(PT p1,PT p2)

{

PT mid;

int v;

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;

}

}

void drawwindow()

{

setcolor(RED);

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);

}

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);

}

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;

}

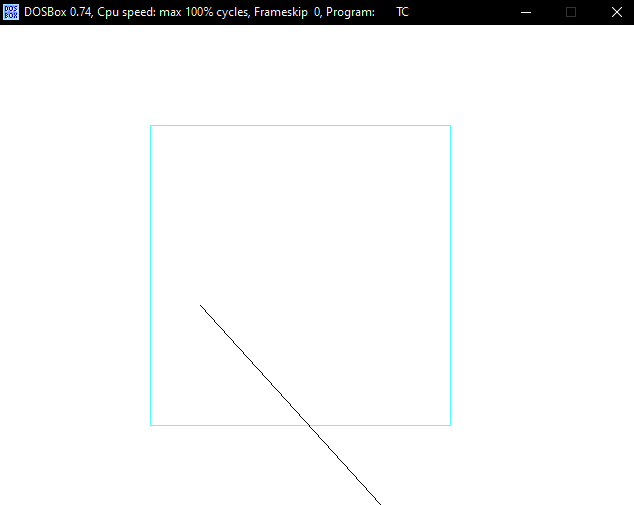
if(flag==0) return(1); return(2);

}

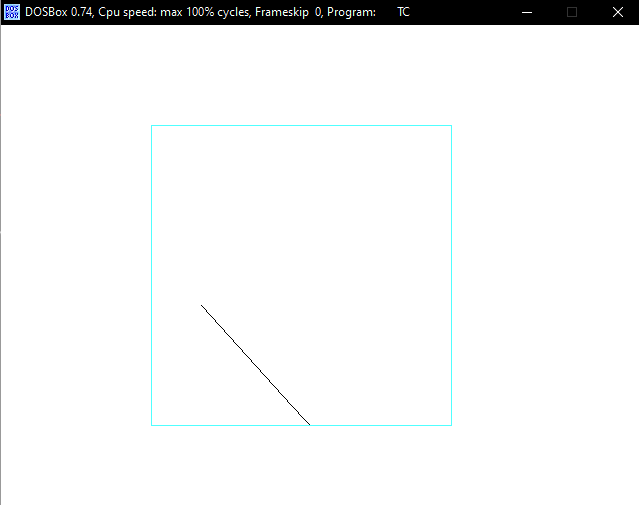
**Output:**



**BEFORE CLIPPING**

****

**AFTER CLIPPING**

****