



### Experiment No. 7

**Aim:** To implement Line Clipping Algorithm: Liang Barsky

**Objective:**

To understand the concept of Liang Barsky algorithm to efficiently determine the portion of a line segment that lies within a specified clipping window. This method is particularly effective for lines predominantly inside or outside the window.

**Theory:**

This Algorithm was developed by Liang and Barsky. It is used for line clipping as it is more efficient because it uses more efficient parametric equations to clip the given line.

These parametric equations are given as:

$$x = x_1 + tdx$$

$$y = y_1 + tdy, 0 \leq t \leq 1$$

Where  $dx = x_2 - x_1$  &  $dy = y_2 - y_1$

**Algorithm**

1. Read 2 endpoints of line as  $p_1 (x_1, y_1)$  &  $p_2 (x_2, y_2)$ .
2. Read 2 corners (left-top & right-bottom) of the clipping window as  $(x_{wmin}, y_{wmin}, x_{wmax}, y_{wmax})$ .
3. Calculate values of parameters  $p_i$  and  $q_i$  for  $i = 1, 2, 3, 4$  such that

$$p_1 = -dx, q_1 = x_1 - x_{wmin}$$

$$p_2 = dx, q_2 = x_{wmax} - x_1$$

$$p_3 = -dy, q_3 = y_1 - y_{wmin}$$

$$p_4 = dy, q_4 = y_{wmax} - y_1$$

4. if  $p_i = 0$  then line is parallel to  $i$ th boundary

if  $q_i < 0$  then line is completely outside boundary so discard line



else, check whether line is horizontal or vertical and then check the line endpoints with the corresponding boundaries.

5. Initialize t1 & t2 as

$t1 = 0$  &  $t2 = 1$



6. Calculate values for  $q_i/p_i$  for  $i = 1, 2, 3, 4$ .

7. Select values of  $q_i/p_i$  where  $p_i < 0$  and assign maximum out of them as t1.

8. Select values of  $q_i/p_i$  where  $p_i > 0$  and assign minimum out of them as t2.

9. if ( $t1 < t2$ )

{

$xx1 = x1 + t1dx$

$xx2 = x1 + t2dx$

$yy1 = y1 + t1dy$

$yy2 = y1 + t2dy$

line ( $xx1, yy1, xx2, yy2$ )

}

10. Stop.

### Program:

```
#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=120;
y1=120;
x2=300;
y2=300;
xmin=100;
ymin=100;
xmax=250;
ymax=250;
initgraph(&gd,&gm," ");
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);  
}  
}
```



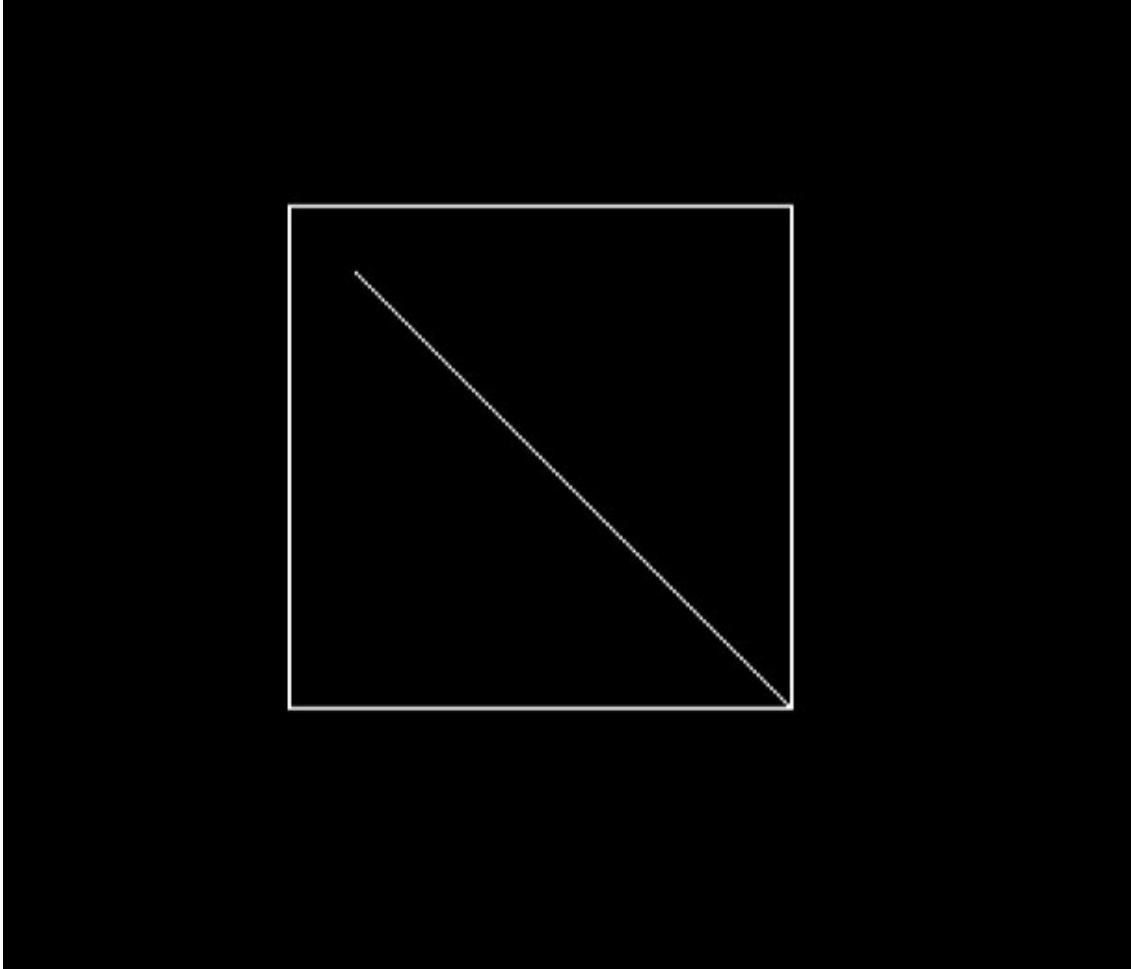
```
}  
}  
}  
t1=0;  
t2=1;  
for(i=0;i<4;i++)  
{  
temp=q[i]/p[i];  
if(p[i]<0)  
{  
if(t1<=temp)  
t1=temp;  
}  
else  
{  
if(t2>temp)  
t2=temp;  
}  
}  
if(t1<t2)  
{  
xx1 = x1 + t1 * p[1];  
xx2 = x1 + t2 * p[1];  
yy1 = y1 + t1 * p[3];  
yy2 = y1 + t2 * p[3];  
line(xx1,yy1,xx2,yy2);  
}  
delay(5000);
```



```
closegraph();
```

```
}
```

**Output:**



**Conclusion:**