```
/*WAP to implement BLA algorithm in Cpp*/
//Bresenham Line Drawing Algorithm
#include <iostream>//line(x1,y1,x2,y2)
#include <cmath>
#include <graphics.h>
using namespace std;
int x_1,x_2,y_1,y_2,p_n,dx,dy,add_y;
float m;
void line_plot_m_s() // del_y = y2-y1 , del_x = x2 -x1 , p(0)=2*del_y-del_x, if
p(n)>0 p(n+1)=p(n)+2*del_y-2*del_x, if p(n)<=0 p(n+1)=p(n)+2*del_y
{
   if (p_n<0)
   {
      p_n=p_n+2*dy;
      x_1++;           // changhe x_1 only
   }
   else
   {
      p_n=p_n+2*dy-2*dx;
      x_1++;         //change both x_1 and y_1
      y_1=y_1+add_y;
   }
   putpixel(x_1,y_1,GREEN);
}
void line_plot_m_l() // del_y = y2-y1 , del_x = x2 -x1 , p(0)=2*del_x-del_y, if
p(n)>0 p(n+1)=p(n)+2*del_x-2*del_y, if p(n)<=0 p(n+1)=p(n)+2*del_x
{
   if (p_n<0)
   {
      p_n=p_n+2*dx;
      y_1=y_1+add_y;           //change y_1 only
   }
   else
   {
      p_n=p_n+2*dx-2*dy;
      x_1++;         //change both x_1 and y_1
      y_1=y_1+add_y;
```

```
   }
   putpixel(x_1,y_1,GREEN);
}
int main()
{
   int i;
   while(1)
   {
      cout<<"\n\n\n\t\t\t\t\t1366*768 ";
      cout<<"\n\n\n\t\t Enter line coordinates (x1,y1), (x2,y2) with in range (0,0)
to (1365,767)";
      cout<<"\n\n Enter (x1,y1)";
      cout<<"\n Enter x1: ";
      cin>>x_1;
      cout<<" Enter y1: ";
      cin>>y_1;
      cout<<"\n\n Enter (x2,y2)";
      cout<<"\n Enter x2: ";
      cin>>x_2;
      cout<<" Enter y2: ";
      cin>>y_2;
      initwindow(1366,768);
      for(i=0; i<=1365; i++) // creates white background
      {
         line(0,i,1365,i);
      }
      //setcolor(GREEN);
      //line(x_1+50,y_1+50,x_2+50,y_2+50);
      if (x_2==x_1)
      {
         if (y_2<y_1)
         {
            y_1=y_1+y_2;
            y_2=y_1-y_2;
            y_1=y_1-y_2;
         }
         while(y_1<y_2)//small slope  |m|=1/0
```

```
   {
      putpixel(x_1,y_1,GREEN);
      y_1++;
   }
   getch();
   break;
}
m=(y_2-y_1)/(x_2-x_1);
if (x_2<x_1)//swap
{
   x_1=x_1+x_2;
   x_2=x_1-x_2;
   x_1=x_1-x_2;
   y_1=y_1+y_2;
   y_2=y_1-y_2;
   y_1=y_1-y_2;
}
dx=abs(x_2-x_1);
dy=abs(y_2-y_1);
if (y_2<y_1)
{
   add_y=-1;
}
else
{
   add_y=1;
}
putpixel(x_1,y_1,GREEN);
if (fabs(m)<1)// small slope |m|<1
{
   p_n=2*dy-dx;
   while(x_1<x_2)//small slope  |m|<1
   {
      line_plot_m_s();
   }
}
else
```

```
      {
         p_n=2*dx-dy;
         while(x_1<x_2)// large slope |m|=>1
         {
            line_plot_m_l();
         }
      }
      getch();
      closegraph();
   }
   return 0;
}
/*
Test lines (x1,y1,x2,y2,slope)
(50,60,1200,600,0.46)
(1200,600,50,60,0.46)
(50,600,1200,70,-0.46)
(1200,70,50,600,-0.46)
(50,60,600,700,1.16)
(600,700,50,60,1.16)
(70,600,400,60,-1.63)
(400,60,70,600,-1.63)
(50,50,70,70,1)
(70,70,50,50,1)
(70,30,50,50,-1)
(50,50,70,30,-1)
(80,70,900,70,0)
(900,70,80,70,0)
(80,70,80,700,1/0)
(80,700,80,70,-1/0)
*/


/*WAP to implement BLA algorithm in Cpp*/
//Bresenham Line Drawing Algorithm
#include<GL/gl.h>
#include<GL/glu.h>
```

```
#include<GL/glut.h>
#include<iostream>
#include<math.h>
using namespace std;
void display();
void reshape(int,int);
void draw();
void takeData();
float X1,X2,Y1,Y2;
void init(){

   glClearColor(0,0,0,1.0);

}

int main(int argc, char**argv){

   takeData();
   glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_RGB);

   glutInitWindowPosition(200,100);
   glutInitWindowSize(500,500);

   glutCreateWindow("BLA");
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   init();

   glutMainLoop();

}

void display(){
   glClear(GL_COLOR_BUFFER_BIT);
   glLoadIdentity();
```

```
    //axis display
    glPointSize(1);
    glColor3f(1,1,1);
    glBegin(GL_LINES);
    glVertex2f(-250,0);
    glVertex2f(250,0);
    glVertex2f(0,-250);
    glVertex2f(0,250);

    glEnd();

    //draw
    glBegin(GL_POINTS);

    draw();

    glEnd();

    glFlush();

}

void reshape(int w, int h){

    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    gluOrtho2D(-250,250,-250,250);
    glMatrixMode(GL_MODELVIEW);

}
```

```
// Main BLA Code
void draw(){

   glColor3f(1,1,1);


   float x1,x2,y1,y2,step,mx,my,dx,dy,temp,p,a;
   x1=X1;
   x2=X2;
   y1=Y1;
   y2=Y2;
   if(((x2-x1)<0&&(y2-y1)>0) || ((x2-x1)>0&&(y2-y1)<0))
      a=-1;
   else
      a=1;

   dx=abs(x2-x1);
   dy=abs(y2-y1);
   if(dy<dx){
      if(x1>x2){
         temp=x1;
         x1=x2;
         x2=temp;
         temp=y1;
         y1=y2;
         y2=temp;
      }
      p=2*dy -dx;
      //cout<<x1<<"\t"<<y1<<endl;
      glVertex2f(x1,y1);
      while((x1)<(x2)){
         if(p<0)
            p=p+2*dy;
         else{
            p=p+2*dy-2*dx;
            y1=y1+a;
         }
```

```
        x1=x1+1;
        //cout<<x1<<"\t"<<y1<<endl;
        glVertex2f(x1,y1);
      }
    }
    else{
      if(y1>y2){
        temp=x1;
        x1=x2;
        x2=temp;
        temp=y1;
        y1=y2;
        y2=temp;
      }
      p=2*dx -dy;
      //cout<<x1<<"\t"<<y1<<endl;
      glVertex2f(x1,y1);
      while((y1)<(y2)){
        if(p<0)
          p=p+2*dx;
        else{
          p=p+2*dx-2*dy;
          x1=x1+a;
        }
        y1=y1+1;
        //cout<<x1<<"\t"<<y1<<endl;
        glVertex2f(x1,y1);
      }
    }

}


void takeData(){

    cout<<"enter initial point: ";
    cin>>X1>>Y1;
```

```
    cout<<"enter final point: ";
    cin>>X2>>Y2;

}
```