//DDA Line Drawing Algorithm

```
#include<GL/glut.h>
#include<stdlib.h>
#include<iostream>
using namespace std;
float x1,x2,y1,y2;

void simpleline();
void DashedLine();
void DottedLine();

float dy,dx,length,x,y,k,Xin,Yin;
void display()
{
	cout<<"\nSimple Line\n";
	simpleline();
	cout<<"\nDashed Line\n";
	DashedLine();
	cout<<"\nDotted Line\n";
	DottedLine();
}

void simpleline()
{
	cout<<"Enter the value of x1 :";
	cin>>x1;
	cout<<"Enter the value of y1 : ";
	cin>>y1;
	cout<<"Enter the value of x2 : ";
	cin>>x2;
	cout<<"Enter the value of y2 : ";
	cin>>y2;


	dx=x2-x1;
	dy=y2-y1;

	if(abs(dx)> abs(dy))
```

```
	{
		length = abs(dx);
	}
	else
		length = abs(dy);

	Xin = dx/length;
	Yin = dy/length;

	x=x1;
	y=y1;
	glBegin(GL_POINTS);
	glVertex2i(x,y);
	glEnd();

	for(k=1 ;k<=length;k++)
	{
		x= x + Xin;
		y= y + Yin;

		glBegin(GL_POINTS);
		glVertex2i(x,y);
		glEnd();
	}
	glFlush();
}
void DashedLine()
{
	cout<<"Enter the value of x1 :";
	cin>>x1;
	cout<<"Enter the value of y1 : ";
	cin>>y1;
	cout<<"Enter the value of x2 : ";
	cin>>x2;
	cout<<"Enter the value of y2 : ";
	cin>>y2;

	dx = x2 - x1;
	dy = y2 - y1;
```

```
	if (abs(dx) > abs(dy))
	{
		length = abs(dx);
	}
	else
	length = abs(dy);

	Xin = dx /length;
	Yin = dy /length;
	x = x1;
	y = y1;
	glBegin(GL_POINTS);
	glVertex2i(x,y);
	for (int i = 0; i <= length; i++)
	{
		x = x + Xin;
		y = y + Yin;
		if (i % 5 == 0)
		{
			glVertex2i(x,y);
		}
	}
	glEnd();
	glFlush();
}
void DottedLine()
{
	cout<<"Enter the value of x1 :";
	cin>>x1;
	cout<<"Enter the value of y1 : ";
	cin>>y1;
	cout<<"Enter the value of x2 : ";
	cin>>x2;
	cout<<"Enter the value of y2 : ";
	cin>>y2;

	dx = x2 - x1;
	dy = y2 - y1;
```

```
	if (abs(dx) > abs(dy))
	{
		length = abs(dx);
	}
	else
	length = abs(dy);

	Xin = dx /length;
	Yin = dy /length;
	x = x1;
	y = y1;
	glBegin(GL_POINTS);
	glVertex2i(x,y);
	for (int i = 0; i <= length; i++)
	{
		x = x + Xin;
		y = y + Yin;
		if (i % 2 == 0)
		{
			glVertex2i(x,y);
		}
	}
	glEnd();
	glFlush();
}


void init(void)
{
	glClearColor(1.0,0.0,0.0,0.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(-100,100,-100,100);
}

int main(int argc, char** argv)
{
	glutInit(&argc, argv);
```

```
	glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize (500, 500);
	glutInitWindowPosition (500,500);
	glutCreateWindow ("DDA Line Algo");
	init();
	glutDisplayFunc(display);
	glutMainLoop();

	return 0;
}
```