----------------------------------------------------------------------

Write C++/Java program for line drawing using DDA or Bresenhams algorithm with patterns such as solid, dotted, dashed, dash dot and thick.

----------------------------------------------------------------------

#include "line\_types.h"

#include "ui\_line\_types.h"

#include <QPainter>

#include<QtGui>

#include<math.h>

Line\_types::Line\_types(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::Line\_types)

{

ui->setupUi(this);

}

Line\_types::~Line\_types()

{

delete ui;

}

void Line\_types::changeEvent(QEvent \*e)

{

QMainWindow::changeEvent(e);

switch (e->type()) {

case QEvent::LanguageChange:

ui->retranslateUi(this);

break;

default:

break;

}

}

void Line\_types :: paintEvent(QPaintEvent \*event)

{

QPainter painter(this);

painter.setPen(QPen((Qt::red),5));

int wy,i=0;

x1=ui->x1->toPlainText().toInt();

y1=ui->y1->toPlainText().toInt();

x2=ui->x2->toPlainText().toInt();

y2=ui->y2->toPlainText().toInt();

th=ui->thickness->toPlainText().toInt();

switch (flag)

{

case 1:dashed(x1,y1,x2,y2);

break;

case 2:dotted(x1,y1,x2,y2);

break;

case 3:dashdot(x1,y1,x2,y2);

break;

case 4:draw\_bresen(x1,y1,x2,y2);

break;

case 5:

wy = (th-1)\*(sqrt(pow((x2-x1),2) + pow((y2-y1),2)))/2/abs(x2-x1);

draw\_bresen(x1,y1,x2,y2);

while(i<=wy)

{

draw\_bresen(x1,y1+i,x2,y2+i);

draw\_bresen(x1,y1-i,x2,y2-i);

i++;

}

break;

}

}

void Line\_types :: dashed (int x1,int y1,int x2,int y2)

{

QPainter painter(this);

painter.setPen(QPen((Qt::red),1));

//painter.drawRect(viewport()->rect());

int i=0;

int e,x,y,dx,dy;

dx=abs(x2-x1);

dy=abs(y2-y1);

x=x1;

y=y1;

i=1;

e=2\*dy-dx;

do

{

if(i%16<=8)

{

painter.drawPoint(x,y);

}

while(e>=0)

{

y=y+1;

e=e-2\*dx;

}

x=x+1;

e=e+2\*dy;

i=i+1;

}while(i<=dx);

}

void Line\_types :: dotted (int x1,int y1,int x2,int y2)

{

QPainter painter(this);

painter.setPen(QPen((Qt::red),1));

int i=0;

int e,x,y,dx,dy;

dx=abs(x2-x1);

dy=abs(y2-y1);

x=x1;

y=y1;

i=1;

e=2\*dy-dx;

do

{

if(i%4==2)

{

painter.drawPoint(x,y);

}

while(e>=0)

{

y=y+1;

e=e-2\*dx;

}

x=x+1;

e=e+2\*dy;

i=i+1;

}while(i<=dx);

}

void Line\_types :: dashdot(int x1,int y1,int x2,int y2)

{

QPainter painter(this);

painter.setPen(QPen((Qt::red),1));

int i=0;

int e,x,y,dx,dy;

dx=abs(x2-x1);

dy=abs(y2-y1);

x=x1;

y=y1;

i=1;

e=2\*dy-dx;

do

{

if(i%16>=10 && i%16<=13)

{

painter.drawPoint(x,y);

}

if(i%16<8)

{

painter.drawPoint(x,y);

}

while(e>=0)

{

y=y+1;

e=e-2\*dx;

}

x=x+1;

e=e+2\*dy;

i=i+1;

}while(i<=dx);

}

void Line\_types :: draw\_bresen(int x1,int y1,int x2,int y2)

{ QPainter painter(this);

painter.setPen(QPen((Qt::blue),1));

int dx,d,dy,x,y,xi,yi,i=0;

dx=abs(x2-x1);

dy=abs(y2-y1);

x=x1;

y=y1;

if(dx>=dy)//gentle slope

{

d=2\*dy-dx;

xi=1;

yi=1;

if(x2<x1)

xi=-1;

if(y2<y1)

yi=-1;

while(i++<dx) //gentle slope

{

if(d>=0)

{

d=d+2\*dy-2\*dx;

painter.drawPoint(x=x+xi,y=y+yi);

continue;

}

if(d<0)

{

d=d+2\*dy;

painter.drawPoint(x=x+xi,y);

}

}

}

if(dy>dx) // steep slope

{

d=2\*dx-dy;

xi=1;

yi=1;

if(x2<x1)

xi=-1;

if(y2<y1)

yi=-1;

while(i++<dy) //steep slope

{

if(d>=0)

{

d=d+2\*dx-2\*dy;

painter.drawPoint(x=x+xi,y=y+yi);

continue;

}

if(d<0)

{

d=d+2\*dx;

painter.drawPoint(x,y=y+yi);

}

}

}

}

void Line\_types::on\_dashed\_clicked()

{

flag=1;

make\_invisible();

update();

}

void Line\_types::on\_dotted\_clicked()

{

flag=2;

make\_invisible();

update();

}

void Line\_types::on\_dashdot\_clicked()

{

flag=3;

make\_invisible();

update();

}

void Line\_types::on\_solid\_clicked()

{

flag=4;

make\_invisible();

update();

}

void Line\_types::on\_thick\_clicked()

{

flag=5;

make\_invisible();

update();

}

void Line\_types::make\_invisible()

{

ui->label->setVisible(false);

ui->label\_2->setVisible(false);

ui->label\_3->setVisible(false);

ui->label\_4->setVisible(false);

ui->label\_5->setVisible(false);

ui->x1->setVisible(false);

ui->y1->setVisible(false);

ui->x2->setVisible(false);

ui->y2->setVisible(false);

ui->thickness->setVisible(false);

ui->dashed->setVisible(false);

ui->dotted->setVisible(false);

ui->dashdot->setVisible(false);

ui->solid->setVisible(false);

ui->thick->setVisible(false);

}

void Line\_types::on\_back\_clicked()

{

ui->label->setVisible(true);

ui->label\_2->setVisible(true);

ui->label\_3->setVisible(true);

ui->label\_4->setVisible(true);

ui->label\_5->setVisible(true);

ui->x1->setVisible(true);

ui->y1->setVisible(true);

ui->x2->setVisible(true);

ui->y2->setVisible(true);

ui->thickness->setVisible(true);

ui->dashed->setVisible(true);

ui->dotted->setVisible(true);

ui->dashdot->setVisible(true);

ui->solid->setVisible(true);

ui->thick->setVisible(true);

}

**OUTPUT:**

Line of thickness = 5 :

