**Assignment no: 04**

**//Aim:Write a C++ class for a circle drawing inheriting line class.**

#include<graphics.h>

#include<iostream>

#include<stdlib.h>

#include<math.h>

using namespace std;

class dline //base class

{

protected: int x1,y1,x2,y2, colour;

public:

dline()

{

x1=0; y1=0; x2=0, y2=0;

}

void setcolor(int color)

{

colour =color;

}

void setline(float x1,float y1, float xx,float yy)

{

x1=x1; y1=y1; x2=xx, y2=yy;

}

void drawl(float x1,float y1, float xx,float yy) //Simple DDA Line

{

float x,y,dx,dy,len;

int i;

dx=abs(x2-x1);

dy=abs(y2-y1);

if(dx >= dy)

{

len=dx;

}

else

{

len=dy;

}

dx=(x2-x1)/len;

dy=(y2-y1)/len;

x = x1 + 0.5;

y = y1 + 0.5;

i=1;

while(i<=len)

{

putpixel(x,y,colour);

x = x + dx;

y = y + dy;

i = i + 1;

}

}

};

class mcircle:public dline

{

private: int x0,y0;

public:

mcircle():dline()

{

x0=0;y0=0;

}

void drawc(float x1, float y1, int r)

{

int i, x, y;

float d;

x=0, y=r;

d = 1.25 - r; //decision variabel

dline::setline(x,y,x,y);

do

{

dline::drawl(x1+x, y1+y,x1+x, y1+y);

dline::drawl(x1+y, y1+x,x1+y, y1+x);

dline::drawl(x1+y, y1-x,x1+y, y1-x);

dline::drawl(x1+x, y1-y,x1+x, y1-y);

dline::drawl(x1-x, y1-y,x1-x, y1-y);

dline::drawl(x1-y, y1-x,x1-y, y1-x);

dline::drawl(x1-y, y1+x,x1-y, y1+x);

dline::drawl(x1-x, y1+y,x1-x, y1+y);

if(d<0)

{

x = x + 1;

d = d + (2\*x) + 3;

}

else

{

x = x + 1;

y = y - 1;

d = d + (2\*x-2\*y) + 3;

}

}while(x<=y);

}

};

int main()

{

int gd=DETECT,gm=VGAMAX;

int i, x, y, r, xmax,ymax,xmid,ymid;

initgraph(&gd,&gm,NULL);

dline l;

mcircle c;

cout<<"\n Enter x: "; cin>>x;

cout<<"\n Enter y: "; cin>>y;

cout<<"\n Enter radius: "; cin>>r;

l.setcolor(15);

c.drawc(x,y,r);

delay(3000);

getch();

closegraph();

return 0;

}

...............................................................................................................................................................................................

#include<graphics.h>

#include<iostream>

#include<stdlib.h>

#include<math.h>

using namespace std;

class dline //base class

{

protected: int x0,y0,x1,y1,x2,y2, colour;

public:

dline()

{

x1=0; y1=0; x2=0, y2=0;

}

void setcolor(int color)

{

colour =color;

}

void setoff1(int xx,int yy)

{

x0=xx;

y0=yy;

}

void setline(float x1,float y1, float xx,float yy)

{

x1=x1+x0; y1=y0-y1; x2=xx+x0, y2=y0-yy;

}

void drawl(float x1,float y1, float xx,float yy) //Simple DDA Line

{

float x,y,dx,dy,len;

int i;

dx=abs(x2-x1);

dy=abs(y2-y1);

if(dx >= dy)

{

len=dx;

}

else

{

len=dy;

}

dx=(x2-x1)/len;

dy=(y2-y1)/len;

x = x1 + 0.5;

y = y1 + 0.5;

putpixel(x,y,colour);

x = x + dx;

y = y + dy;

}

};

class mcircle:public dline

{

private: int x0,y0;

public:

mcircle():dline()

{

x0=0;y0=0;

}

void setoff(int xx,int yy)

{

x0=xx;

y0=yy;

}

void drawc(float x1, float y1, int r)

{

int i, x, y;

float d;

x=0, y=r;

d = 1.25 - r; //decision variabel

dline::setline(x,y,x,y);

do

{

dline::drawl(x1+x0+x, y0-y1+y,x1+x0+x, y0-y1+y);

dline::drawl(x1+x0+y, y0-y1+x,x1+x0+y, y0-y1+x);

dline::drawl(x1+x0+y, y0-y1-x,x1+x0+y, y0-y1-x);

dline::drawl(x1+x0+x, y0-y1-y,x1+x0+x, y0-y1-y);

dline::drawl(x1+x0-x, y0-y1-y,x1+x0-x, y0-y1-y);

dline::drawl(x1+x0-y, y0-y1-x,x1+x0-y, y0-y1-x);

dline::drawl(x1+x0-y, y0-y1+x,x1+x0-y, y0-y1+x);

dline::drawl(x1+x0-x, y0-y1+y,x1+x0-x, y0-y1+y);

if(d<0)

{

x = x + 1;

d = d + (2\*x) + 3;

}

else

{

x = x + 1;

y = y - 1;

d = d + (2\*x-2\*y) + 3;

}

}while(x<=y);

}

};

int main()

{

int gd=DETECT,gm=VGAMAX;

int i, x, y, r, xmax,ymax,xmid,ymid;

char a;

initgraph(&gd,&gm,NULL);

dline l;

mcircle c;

xmax = getmaxx();

ymax = getmaxy();

xmid = xmax /2;

ymid = ymax /2;

line(xmid,0,xmid,ymax); //Y co-ordinate

line(0,ymid,xmax,ymid); //X co-ordinate

do

{

cout<<"\n Enter x: "; cin>>x;

cout<<"\n Enter y: "; cin>>y;

cout<<"\n Enter radius: "; cin>>r;

c.setoff(xmid, ymid);

l.setoff1(xmid, ymid);

l.setcolor(15);

c.drawc(x,y,r);

cout<<"\nDO U Want To Continue y OR n: ";

cin>>a;

}while(a!='n');

delay(3000);

getch();

closegraph();

return 0;

}