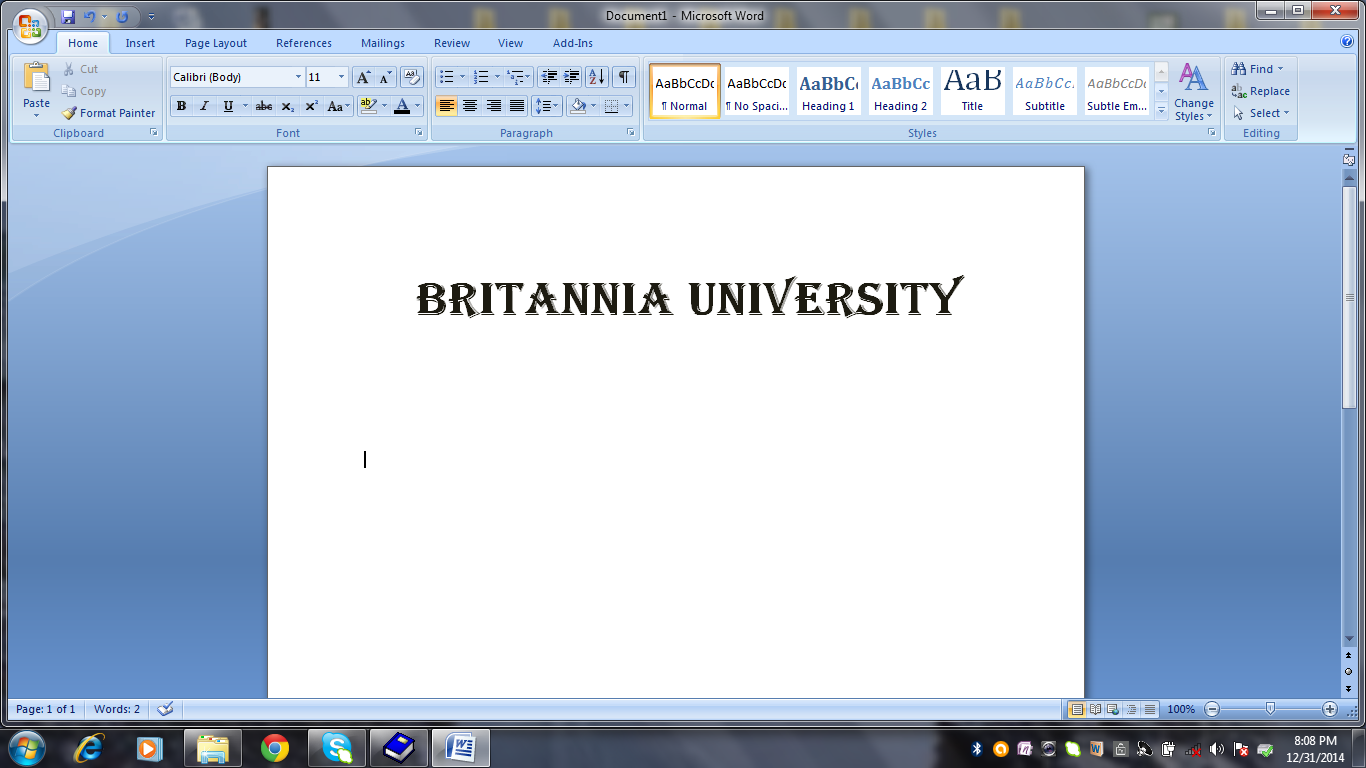
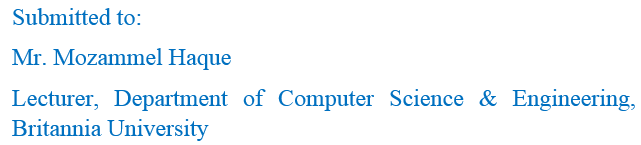


****

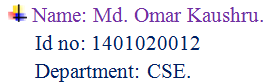
**Course title:** Computer Graphics.

**Report on:** All lab assignment.



Date of Submission: 8th May, 2017

Submitted by:

****

* Bresenham’s Line algorithm.

#include<iostream>

#include<graphics.h>

**using** **namespace** std;

**int** main()

{

initwindow(800,800);

**int** x,y,d,dx,dy,x1,y1,x2,y2;

cin>>x1>>y1>>x2>>y2;

x=x1;

y=y1;

dx=x2-x1;

dy=y2-y1;

d=2\*dy-dx;

putpixel(x,y,7);

**while**(x<x2)

{

x++;

**if**(d<0)

d=d+2\*dy;

**else**

{

y++;

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

}

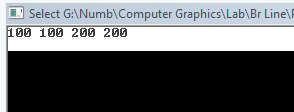
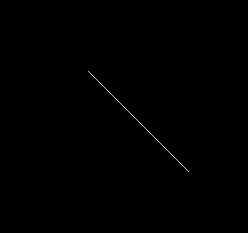
putpixel(x,y,7);

}

getch();

}

**Sample Input/Output:**

* Bresenham’s circle algorithm.

#include<iostream>

#include<graphics.h>

**using** **namespace** std;

**int** main()

{

initwindow(800,800);

**int** x1=10,y1=20,x2=100,y2=20;

**int** r=70;

**int** x=0,y=r, h=50, k=50, d;

d=3-2\*y;

**while**(x<=y)

{

putpixel(x+h,y+k,7);

**if**(d<0)

d=d+4\*x+6;

**else**

{

d=d+4\*(x-y)+10;

y--;

}

x++;

}

getch();

}

**Sample Input/Output:**



* Moving line.

#include<iostream>

#include<graphics.h>

**using** **namespace** std;

**int** main()

{

initwindow(800,800);

**int** x1=10,y1=20,x2=100,y2=20;

**for**(; ;)

{

cleardevice();

x1=x1+20;

x2=x2+20;

line(x1,y1,x2,y2);

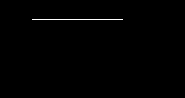
delay(400);

}

getch();

}

**Sample input/output:**



* DDA algorithm.

#include<iostream>

#include<math.h>

#include<stdio.h>

#include<graphics.h>

**using** **namespace** std;

**int** main()

{

initwindow(800,800);

**int** x1,y1,x2,y2,i,j;

**double** x,y,m;

cin>>x1>>y1>>x2>>y2;

m=(y2-y1)/**double**(x2-x1);

**if**(m<=1)

{

y=y1;

**for**(i=x1; i<=x2; i++)

{

j=floor(y+0.5);

putpixel(i,j,7);

y=y+m;

}*///end of for loop*

}*//end of if*

**else**

{

x=x1;

**for**(j=y1; j<=y2; j++)

{

i=floor(x+.05);

putpixel(i,j,7);

x=x+(1/m);

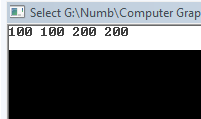
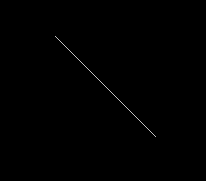
}

}

getch();

}

**Sample input/output:**

* Magnify a rectangle.

#include <math.h>

#include <iostream>

#include<graphics.h>

#define PI 3.14159265

**using** **namespace** std;

**int** main()

{

**int** i,j,kk,fixed;

**double** points[3][3],solution[3][3],S\_ab\_p[3][3],a,b,h,k;

**for**(i=0; i<3; i++)

{

**for**(j=0; j<2; j++)

cin>>points[j][i];

}

**for**(j=0; j<3; j++)

points[2][j]=1;

cout<<"Enter a and b :";

cin>>a>>b;

cout<<"which point you want keep fixed :";

cin>>fixed;

initwindow(800,800);

line(points[0][0],points[1][0],points[0][2],points[1][2]);

line(points[0][0],points[1][0],points[0][1],points[1][1]);

line(points[0][1],points[1][1],points[0][2],points[1][2]);

h=points[0][fixed-1],k=points[1][fixed-1];

S\_ab\_p[0][0]=a,S\_ab\_p[0][1]=0,S\_ab\_p[0][2]=-(a\*h)+h;

S\_ab\_p[1][0]=0,S\_ab\_p[1][1]=b,S\_ab\_p[1][2]=-(b\*k)+k;

S\_ab\_p[2][0]=0,S\_ab\_p[2][1]=0,S\_ab\_p[2][2]=1;

**for**(i=0 ; i<3 ; i++)

{

**for**(j=0 ; j<3; j++)

{

solution[i][j] = 0 ;

**for**(kk=0 ; kk<3 ; kk++)

{

solution[i][j] = solution[i][j] + (( S\_ab\_p[i][kk] \* points[kk][j] ));

}

}

}

**for**(i=0; i<3; i++)

{

**for**(j=0; j<3; j++)

cout<<solution[i][j]<<" ";

cout<<endl;

}

delay(1500);

line(solution[0][0],solution[1][0],solution[0][2],solution[1][2]);

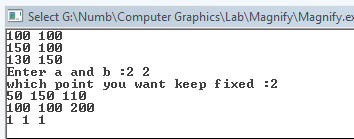
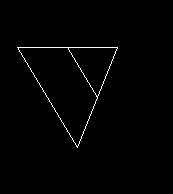
line(solution[0][0],solution[1][0],solution[0][1],solution[1][1]);

line(solution[0][1],solution[1][1],solution[0][2],solution[1][2]);

getch();

}

**Sample Input/Output:**

* Rotate a triangle.

#include <math.h>

#include <iostream>

#include<graphics.h>

**using** **namespace** std;

#define PI 3.14159265

**int** main()

{

**int** i,j,k;

**double** points[3][3],angel,solution[3][3],R\_theta[3][3];

cin>>angel;

**for**(i=0; i<3; i++)

{

**for**(j=0; j<2; j++)

cin>>points[j][i];

}

**for**(j=0; j<3; j++)

points[2][j]=1;

R\_theta[0][0]=cos(angel\*PI/180),R\_theta[0][1]= - (sin(angel\*PI/180)),R\_theta[0][2]=0;

R\_theta[1][0]=sin(angel\*PI/180),R\_theta[1][1]=cos(angel\*PI/180),R\_theta[1][2]=0;

R\_theta[2][0]=0,R\_theta[2][1]=0,R\_theta[2][2]=1;

**for**(i=0 ; i<3 ; i++)

{

**for**(j=0 ; j<3; j++)

{

solution[i][j] = 0 ;

**for**(k=0 ; k<3 ; k++)

{

solution[i][j] = solution[i][j] + (R\_theta[i][k] \* points[k][j]);

}

}

}

for(i=0; i<3; i++)

{

for(j=0; j<3; j++)

{

printf("%.2lf ",solution[i][j]);

}

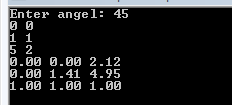
cout<<endl;

}

getch();

}

**Sample input/output:**



* Convex Hull.

#include<graphics.h>

#include <math.h>

#include <iostream>

**using** **namespace** std;

#define PI 3.14159265

**int** top,oStack[20];

**void** push(**int** i)

{

oStack[++top]=i;

}

**int** pop()

{

**return** (oStack[top--]);

}

**double** diterminant(**double** pX,**double** pY,**double** qX,**double** qY,**double** rX,**double** rY)

{

**return**(1\*(qX\*rY-qY\*rX)-pX\*(1\*rY-1\*qY)+ pY\*(1\*rX-1\*qX));

}

**int** main()

{

**int** n,i,j,k,flag,f;

**double** m,angle[20],floTemp,x[20],y[20],temp1,temp2,min;

**while**(cin>>n)

{

top=-1;

min=99999;

**for**(i=0; i<n; i++)

{

cin>>x[i]>>y[i];

**if**(y[i]<=min)

{

min=y[i];

flag=i;

}

}

*///Bringing the minimum value of y coordinate at the first position*

temp1=x[0];

temp2=y[0];

x[0]=x[flag];

y[0]=y[flag];

x[flag]=temp1;

y[flag]=temp2;

*///Calculating angle with respect to the minimum value of Y coordinate*

**for**(i=1; i<n; i++)

{

m=(y[i]-y[0])/(x[i]-x[0]);

**if**(x[i]>x[0] && y[i]>y[0])

floTemp=atan (m) \* 180 / PI;

**else** **if**(x[i]<x[0] && y[i]>y[0])

floTemp=(atan (m) \* 180 / PI)+180;

**else** **if**(x[i]<x[0] && y[i]<y[0])

floTemp=(atan (m) \* 180 / PI)+180;

**else** **if**(x[i]>x[0] && y[i]<y[0])

floTemp=(atan (m) \* 180 / PI)+360;

angle[i-1]=floTemp;

}

*///Sorting X and Y coordinate according to their angel*

**for**(i=0; i<n-1; i++)

{

**for**(j=i+1; j<n-1; j++)

{

**if**(angle[i]>angle[j])

{

*///Sorting Angel*

temp1=angle[i];

angle[i]=angle[j];

angle[j]=temp1;

*///Sorting X coordinate*

temp1=x[i+1];

x[i+1]=x[j+1];

x[j+1]=temp1;

*///Sorting Y coordinate*

temp1=y[i+1];

y[i+1]=y[j+1];

y[j+1]=temp1;

}

}

}*///End of Sorting loop*

cout<<"\n";

**for**(i=0; i<n; i++)

cout<<x[i]<<" "<<y[i]<<endl;

**for**(i=0; i<3; i++)*///pushing first three element*

push(i);

**for**(i=3; i<n; i++)

{

temp1=diterminant(x[oStack[top-1]],y[oStack[top-1]],x[oStack[top]],y[oStack[top]],x[i],y[i]);

**if**(temp1==0)*///if colinear ignore*

{

j=pop();

push(i);

}

**else** **if**(temp1>0)*///left*

{

push(i);

}

**else***///right*

{

j=pop();

push(i);

}

}

cout<<"\n\n";

initwindow(800,800);

**for**(i=0; i<=n; i++)*///Printig the points*

{

putpixel(x[i],y[i],11);

delay(200);

}

delay(900);

**for**(i=0; i<top; i++)

{

line(x[oStack[i]],y[oStack[i]],x[oStack[i+1]],y[oStack[i+1]]);

delay(400);

}

line(x[oStack[i]],y[oStack[i]],x[oStack[0]],y[oStack[0]]);

}

**return** 0;

}

**Sample input/output:**

