

Computer design and manufacturing lab manual

01. BRESENHAM'S ALGORITHM FOR LINE DRAWING.

1. Start.
2. Declare variables $x, y, x_1, y_1, x_2, y_2, p, dx, dy$ and also declare $gdriver=DETECT, gmode$.
3. Initialize the graphic mode with the path location in TC folder.
4. Input the two line end-points and store the left end-points in (x_1, y_1) .
5. Load (x_1, y_1) into the frame buffer; that is, plot the first point put $x=x_1, y=y_1$.
6. Calculate $dx=x_2-x_1$ and $dy=y_2-y_1$, and obtain the initial value of decision parameter p as:
 - a. $p=(2dy-dx)$.
7. Starting from first point (x, y) perform the following test:
8. Repeat step 9 while $(x \leq x_2)$.
9. If $p < 0$, next point is $(x+1, y)$ and $p=(p+2dy)$.
10. Otherwise, the next point to plot is $(x+1, y+1)$ and $p=(p+2dy-2dx)$.
11. Place pixels using `putpixel` at points (x, y) in specified colour.
12. Close Graph.
13. Stop.

WAP TO DRAW A LINE USING MID POINT ALGORITHM OR BRESENHAM'S ALGORITHM.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
    int    x,y,x1,y1,x2,y2,p,dx,dy;
    int gdriver=DETECT,gmode;
    initgraph(&gdriver,&gmode,"C:\\tc\\BGI:");
    printf("\nEnter the x-coordinate of the first point ::");
    scanf("%d",&x1);
    printf("\nEnter the y-coordinate of the first point ::");
    scanf("%d",&y1);
    printf("\nEnter the x-coordinate of the second point ::");
    scanf("%d",&x2);
    printf("\nEnter the y-coordinate of the second point ::");
    scanf("%d",&y2);
    x=x1;
    y=y1;
    dx=x2-x1;
    dy=y2-y1;
    putpixel(x,y,2);
    p=(2dy-dx);
    while(x<=x2)
    {
        if(p<0)
        {
            x=x+1;
```

```
        p=2*x-dx;
    }
    else
    {
        x=x+1;
        y=y+1;
        p=p+2*dy;
    }
    putpixel(x,y,7);
}
getch();
closegraph();
}
```

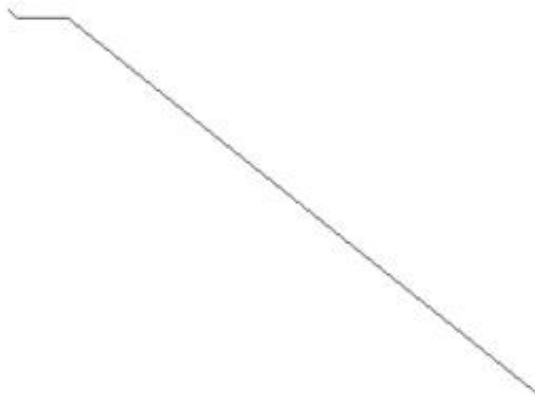
OUTPUT

Enter the x-coordinate of the first point ::180

Enter the y-coordinate of the first point ::250

Enter the x-coordinate of the second point ::500

Enter the y-coordinate of the second point ::600



02. ALGORITHM TO DRAW A LINE USING DDA ALGORITHM.

1. Start.
2. Declare variables $x, y, x_1, y_1, x_2, y_2, k, dx, dy, s, x_i, y_i$ and also declare $gdriver=DETECT, gmode$.
3. Initialise the graphic mode with the path location in TC folder.
4. Input the two line end-points and store the left end-points in (x_1, y_1) .
5. Load (x_1, y_1) into the frame buffer; that is, plot the first point, put $x=x_1, y=y_1$.
6. Calculate $dx=x_2-x_1$ and $dy=y_2-y_1$.
7. If $abs(dx) > abs(dy)$, do $s=abs(dx)$.
8. Otherwise $s=abs(dy)$.
9. Then $x_i=dx/s$ and $y_i=dy/s$.
10. Start from $k=0$ and continuing till $k < s$, the points will be
 - i. $x=x+x_i$.
 - ii. $y=y+y_i$.
11. Place pixels using `putpixel` at points (x, y) in specified colour.
12. Close Graph.
13. Stop.

WAP TO DRAW A LINE USING DDA ALGORITHM.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
    int x,y,x1,x2,y1,y2,k,dx,dy,s,xi,yi;
    int gdriver=DETECT,gmode;
    initgraph(&gdriver,&gmode,"C:\\tc\\bgi:");
    printf("enter first point");
    scanf("%d%d",&x1,&y1);
    printf("enter second point");
    scanf("%d%d",&x2,&y2);
    x=x1;
    y=y1;
    putpixel(x,y,7);
    dx=x2-x1;
    dy=y2-y1;
    if(abs(dx)>abs(dy))
        s=abs(dx);
    else
        s=abs(dy);
    xi=dx/s;
    yi=dy/s;
    x=x1;
    y=y1;
    putpixel(x,y,7);
    for(k=0;k<s;k++)
    {
```

```
        x=x+xi;  
        y=y+yi;  
        putpixel(x,y,7);  
    }  
    getch();  
    closegraph();  
}
```


OUTPUT

```
enter first point100  
200  
enter second point200  
100
```



03. BRESENHAM'S ALGORITHM TO DRAW A CIRCLE.

1. Start.
2. Declare variables x,y,p and also declare gdriver=DETECT,gmode.
3. Initialise the graphic mode with the path location in TC folder.
4. Input the radius of the circle r.
5. Load x=0,y=r,initial decision parameter $p=1-r$.so the first point is (0,r).
6. Repeat Step 7 while $(x < y)$ and increment x-value simultaneously.
7. If $(p > 0)$,do $p=p+2*(x-y)+1$.
8. Otherwise $p=p+2*x+1$ and y is decremented simultaneously.
9. Then calculate the value of the function circlepoints() with p.parameters (x,y).
10. Place pixels using putpixel at points (x+300,y+300) in specified colour in circlepoints() function shifting the origin to 300 on both x-axis and y-axis.
11. Close Graph.
12. Stop.

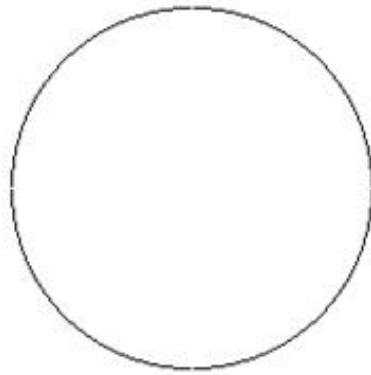
WAP TO DRAW A CIRCLE USING BRESENHAM'S ALGORITHM.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void circlepoints(int,int);
void main()
{
    int x,y,p,r;
    int gdriver=DETECT,gmode;
    initgraph(&gdriver,&gmode,"C:\\tc\\bgi:");
    clrscr();
    printf("enter the radius");
    scanf("%d",&r);
    x=0;y=r;p=1-r;
    while(x<y)
    {
        x++;
        if(p>0)
        {
            p=p+2*(x-y)+1;
            y--;
        }
        else
            p=p+2*x+1;
        circlepoints(x,y);
    }
    getch();
    closegraph();
}
void circlepoints(int x,int y)
```

```
{  
    putpixel(x+300,y+300,8);  
    putpixel(x+300,-y+300,8);  
    putpixel(-x+300,y+300,8);  
    putpixel(-x+300,-y+300,8);  
    putpixel(y+300,x+300,8);  
    putpixel(y+300,-x+300,8);  
    putpixel(-y+300,x+300,8);  
    putpixel(-y+300,-x+300,8); }
```

OUTPUT

enter the radius90



04. ALGORITHM TO DRAW AN ELLIPSE.

1. Start.
2. Initialize the graphic system using initgraph function.
3. Get the input of radius of major and minor arc from the user.
4. Store the values of major and minor arc in an another variable.
5. Square the values of major and minor arc.
6. Calculate decision parameter $P = (\text{square of minor axis} - (\text{square of major axis} * \text{minor axis}) + (0.25 * \text{square of major axis}))$.
7. Put the pixels symmetrically at $(0, \text{length of minor axis})$.
8. while $(2 * (\text{square of minor axis} * x) \leq 2 * (\text{square of major axis} * y))$, repeat steps 9 to step 17.
9. increment x axis by 1.
10. If $P < 0$
11. new $P = (P + (\text{square of minor axis} * \text{square of major axis}) + \text{square of major axis})$
12. Else
13. new $P = (P + (\text{square of minor axis} * x \text{ axis}) - (2 * \text{square of major axis} * y \text{ axis}) + \text{square of minor axis})$.
14. Decrement y by 1.
15. End of step 10 if else structure.
16. Plot symmetric points of ellipse in each quadrant.
17. End of step 8 loop.
18. This will give us ellipse only across minor axis now to draw an ellipse across major axis we proceed further.
19. Get last point of ellipse in 1st quadrant.
20. Initialize $e = \text{square of } (x \text{ axis} + .5)$
21. Initialize $f = \text{square of } (y \text{ axis} - 1)$.
22. Decision parameter $P1 = ((\text{square of minor axis} * e) + (\text{square of major axis} * f) - (\text{square of minor axis} * \text{square of major axis}))$.
23. While $y \text{ axis} \neq 0$ repeat steps 24 to step 32.
24. If $P1 > 0$
25. New $P1 = (P1 + \text{square of major axis} - (2 * \text{square of major axis} * x \text{ axis}))$.

26. Else
27. New $P1 = (P1 + (2 * \text{square of minor axis} * (x \text{ axis} + 1)) - (2 * \text{square of major axis} * (y \text{ axis} - 1))) + \text{square of major axis}$.
28. Increment x axis by 1.
29. End of step 25 if else structure
30. Decrement y axis by 1.
31. Plot symmetric point in all quadrants
32. End of step 23 while loop.
33. Close the graphic system.
34. Stop.

WAP TO DRAW AN ELLIPSE USING MID-POINT ELLIPSE DRAWING ALGORITHM.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void ellips(int x,int y);
void completellipse(int r,int g,int u,int v)
{
    float s,k,e,f,x;
    double p1,p2;
    s=r;k=g;
    e=(pow((s+.5),2));
    f=(pow((k-1),2));
    p2=((u*e)+(v*f)-(u*v));
    ellips(s,k);
    while(k>=0)
    {
        if(p2>0)
            p2=(p2+v-(2*v*s));
        else
        {
            p2=(p2+(2*u*(s+1))-(2*v*(k-1))+v);
            s++;
        }
        k--;
        ellips(s,k);
    }
}
```



```

void main()
{
    int gdriver=DETECT,gmode;
    int a,b,x,y;
    long u,v,p1;
    initgraph(&gdriver,&gmode,"C:\\tc\\bgi:");
    printf("\n enter the length of major axis:");
    scanf("\t%d",&a);
    printf("\n enter the length of minor axis:");
    scanf("\t%d",&b);
    x=0;
    y=b;
    u=pow(b,2);
    v=pow(a,2);
    p1=(u-(v*b)+(.25*v));
    ellips(x,y);
    while(2*(u*x)<=2*(v*y))
    {
        x++;
        if(p1<0)
            p1=(p1+(2*u*v)+v);
        else
        {
            p1=(p1+(2*u*x)-(2*v*y)+u);
            y--;
        }
        ellips(x,y);
    }
    completellipse(x,y,u,v);
    getch();
    closegraph();
}

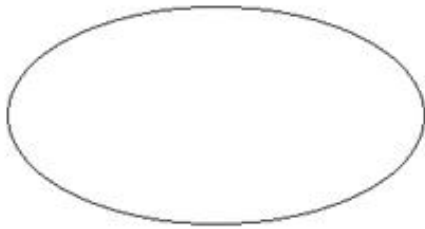
```

```
}  
void ellips(int x,int y)  
{  
    putpixel(x+200,y+200,8);  
    putpixel(-x+200,y+200,8);  
    putpixel(x+200,-y+200,8);  
    putpixel(-x+200,-y+200,8);  
}
```

OUTPUT

enter the length of major axis:100

enter the length of minor axis:50



05. ALGORITHM TO CLIP A LINE.

1. Start.
2. Initialize the graphic system using initgraph function.
3. Get the input of window co ordinates from the user and draw a window.
4. Get the input of line co ordinates from user and draw the line.
5. Calculate the region code of each end point of line using relation given in steps 6 to step
6. Let (x,y) be the co ordinates of end point of line and (x_{min},y_{min}) , (x_{max},y_{max}) be co ordinates of world window
7. If $y - y_{max} = +ve$
8. MSB region code = 1.
9. Else MSB region code = 0.
10. If $y_{min} - y = +ve$
11. Region code = 1.
12. Else Region code = 0.
13. If $x - x_{max} = +ve$
14. Region code = 1.
15. Else Region code = 0.
16. If $x_{min} - x = +ve$
17. LSB Region code = 1.
18. Else LSB Region code = 0.
19. Calculate region code of both end points.
20. Logically and both region code.
21. If Logically anded result is = 0
22. Line is not a clipping candidate.
23. Else.
24. Line is a clipping candidate.
25. Calculate slope of line using formula $slope = (y_2 - y_1) / (x_2 - x_1)$.
26. If line is to be horizontally clipped.
27. New $y = y_{min}$ or y_{max} .
28. New $x = x_1 + ((new\ y - y_1) / slope)$.

29. If line is to be vertically clipped.
30. New $x = x_{\min}$ or x_{\max} .
31. New $y = y_1 + \text{slope} * (\text{new } x - x_1)$.
32. Clip the lines from these intersection points.
33. Display the new line.
34. Close the graphic system.
35. Stop.

WAP TO SHOW LINE CLIPPING.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void storepoints(int,int,int,int,int,int,int,int[]);
void main()
{
    int gdriver=DETECT,gmode;
    int x1,x2,y1,y2,xmax,ymax,xmin,ymin,a[10],b[10],xi1,xi2,yi1,yi2,flag=0;
    float m;
    int i;
    clrscr();

    printf("output");
    printf("\n");
    printf("enter the value of x1,y1,x2,y2:___>");
    scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
    printf("enter the value of xmax,ymax,xmin,ymin:");
    scanf("%d%d%d%d",&xmax,&ymax,&xmin,&ymin);
    storepoints(x2,y2,ymin,ymax,xmax,xmin,b);
    for(i=1;i<=4;i++)
    {
        if(a[i]*b[i]==0)
            flag=1;
        else
            flag=0;
    }
    if(flag==1)
```

```
{
    m=(y2-y1)/(x2-x1);
    xi1=x1;
    yi1=y1;
}
if(a[1]==1)
{
    yi1=ymax;
    xi1=x1+((1/m)*(yi1-y1));
}
else
{
    if(a[2]==1)
    {
        yi1=ymin;
        xi1=x1+((1/m)*(yi1-y1));
    }
}
if(a[3]==1)
{
    xi1=xmax;
    yi1=y1+(m*(xi1-x1));
}
if(a[4]==1)
{
    xi1=xmin;
    yi1=y1+(m*(xi1-x1));
}
else
    if(b[1]==1)
    {
```

```

        yi2=ymax;
        xi2=x2+((1/m)*(yi2-y2));
    }
    else
        if(b[2]==1)
        {
            yi2=ymin;
            xi2=x2+((1/m)*(yi2-y2));
        }
    else
        if(b[3]==1)
        {
            xi2=xmax;
            yi2=y2+((1/m)*(xi2-x2));
        }
    else
        if(b[4]==1)
        {
            xi2=xmin;
            yi2=y2+(m*(xi2-x2));
        }

    clrscr();
    initgraph(&gdriver,&gmode,"c://tc//bgi:");
    rectangle(xmin,ymin,xmax,ymax);
    line(x1,y1,x2,y2);
    delay(5000);
    closegraph();
    clrscr();
    initgraph(&gdriver,&gmode,"c://tc//bgi:");
    line(xi1,yi1,xi2,yi2);
    rectangle(xmin,ymin,xmax,ymax);

```



```
    if(flag==0)
    {
        printf("\n no clipping is required");
    }
    getch();
    closegraph();
}
void storepoints(int x1,int y1,int ymax,int xmax,int xmin,int ymin,int c[10])
{
    if((y1-ymax)>0)
        c[1]=1;
    else
        c[1]=0;
    if((ymin-y1)>0)
        c[2]=1;
    else
        c[2]=0;
    if((x1-xmax)>0)
        c[3]=1;
    else
        c[3]=0;
    if((xmin-x1)>0)
        c[4]=1;
    else
        c[4]=0;
}
```

OUTPUT

enter the value of x1,y1,x2,y2:___>10

10

100

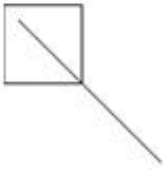
100

enter the value of xmax,ymax,xmin,ymin50

50

0

0



06.WAP TO ROTATE A TRIANGLE ABOUT ORIGIN.

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<process.h>
#include<math.h>
void main()
{
clrscr();
int graphdriver=DETECT,graphmode;
initgraph(&graphdriver,&graphmode,"...\\bgi");

int x,y,x1,a[3][3];
double b[3][3],c[3][3];
cout<<"\n      Enter 1st coordinates of triangle:";
cin>>a[0][0]>>a[1][0];

cout<<"\n      Enter 2nd coordinates of triangle:";
cin>>a[0][1]>>a[1][1];

cout<<"\n      Enter 3rd coordinates of triangle:";
cin>>a[0][2]>>a[1][2];

line(a[0][0],a[1][0],a[0][1],a[1][1]);
line(a[0][1],a[1][1],a[0][2],a[1][2]);
line(a[0][0],a[1][0],a[0][2],a[1][2]);
getch();
cleardevice();
cout<<"\n Enter angle of rotation:\n";
cin>>x;
```

```

b[0][0]=b[1][1]=cos((x*3.14)/180);
b[0][1]=-sin((x*3.14)/180);
b[1][0]=sin((x*3.14)/180);
b[2][2]=1;
b[2][0]=b[2][1]=b[0][2]=b[1][2]= 0;
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{
c[i][j]=0;
for (int k=0; k<3;k++)
{
c[i][j]+=a[i][k]*b[k][j];
}
x1=(c[i][j]+0.5);
a[i][j]=x1;
}
}
cout<<"\n Triangle after rotation is:\n" ;

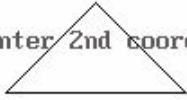
line(a[0][0],a[1][0],a[0][1],a[1][1]);
line(a[0][1],a[1][1],a[0][2],a[1][2]);
line(a[0][0],a[1][0],a[0][2],a[1][2]);

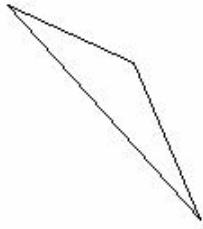
getch();
closegraph();
}

```

OUTPUT

```
Enter 1st coordinates of triangle:100
100
Enter 2nd coordinates of triangle:200
100
Enter 3rd coordinates of triangle:150
50
```





Enter angle of rotation:
30

Triangle after rotation is:

07. PROGRAM TO SCALE THE TRIANGLE

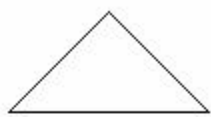
```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd=DETECT, gm;
initgraph(&gd, &gm, "");
cleardevice();
int x1, y1, x2, y2, x3, y3, x4, y4;
float sx, sy;
cout<<"Enter the first coordinates of triangle\n";
cin>>x1>>y1;
cout<<"Enter the second coordinates of triangle\n";
cin>>x2>>y2;
cout<<"Enter the third coordinates of triangle\n";
cin>>x3>>y3;
int poly[8]={x1, y1, x2, y2, x3, y3, x1, y1};
cleardevice();
drawpoly(4, poly);
getch();
cout<<"Enter the scaling factors\n";
cin>>sx>>sy;
x4=sx*x1-x1;
y4=sy*y1-y1;

x1=sx*x1-x4;
y1=sy*y1-y4;
x2=sx*x2-x4;
```

```
y2=sy*y2-y4;  
x3=sx*x3-x4;  
y3=sy*y3-y4;  
poly[0]=x1;  
poly[1]=y1;  
poly[2]=x2;  
poly[3]=y2;  
poly[4]=x3;  
poly[5]=y3;  
poly[6]=x1;  
poly[7]=y1;  
getch();  
cleardevice();  
drawpoly(4,poly);  
getch();  
closegraph();  
}
```


OUTPUT

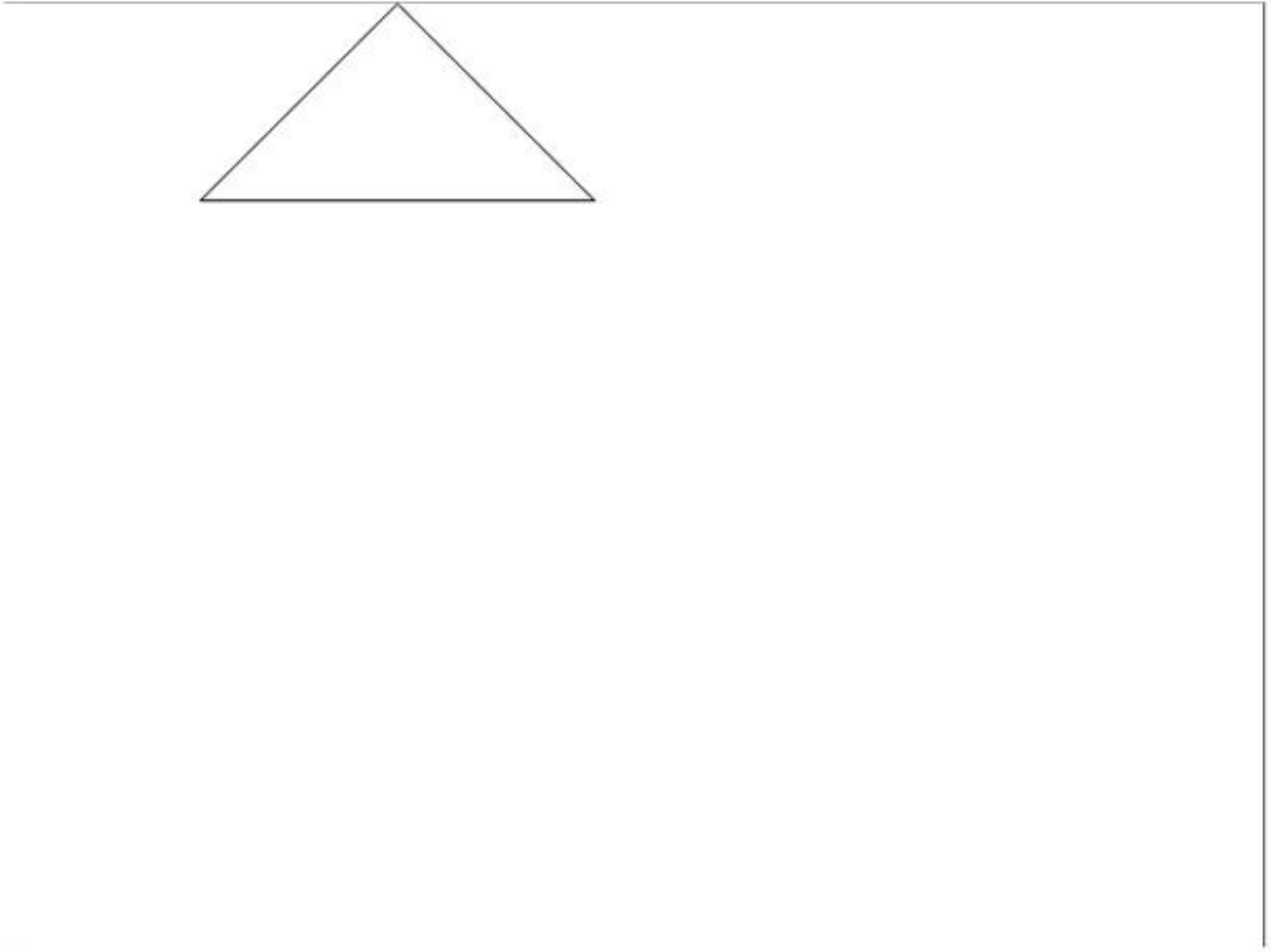
```
Enter the first coordinates of triangle  
100  
100  
Enter the second coordinates of triangle  
200  
100  
Enter the third coordinates of triangle  
150  
50
```



Enter the scaling factors

2

2



08. PROGRAM TO TRANSLATE A TRIANGLE

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<process.h>
#include<math.h>

void main()
{
clrscr();
int graphdriver=DETECT,graphmode;
initgraph(&graphdriver,&graphmode,"...\\bgi");

int x,y,x1,y1,x2,y2,x3,y3;
cout<<"\n      Enter 1st coordinates of triangle:";
cin>>x1>>y1;

cout<<"\n      Enter 2nd coordinates of triangle:";
cin>>x2>>y2;

cout<<"\n      Enter 3rd coordinates of triangle:";
cin>>x3>>y3;

cleardevice();
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x1,y1,x3,y3);
getch();
cleardevice();
```

```
cout<<"\n Enter translatio factors :\n";
```

```
cin>>x>>y;
```

```
x1-=x;
```

```
y1-=y;
```

```
x2-=x;
```

```
y2-=y;
```

```
x3-=x;
```

```
y3-=y;
```

```
cleardevice();
```

```
line(x1,y1,x2,y2);
```

```
line(x2,y2,x3,y3);
```

```
line(x1,y1,x3,y3);
```

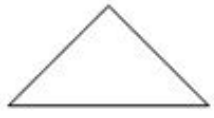
```
getch();
```

```
closegraph();
```

```
}
```

OUTPUT

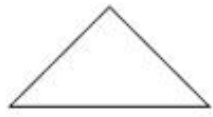
```
100      Enter 1st coordinates of triangle:100
100      Enter 2nd coordinates of triangle:200
50       Enter 3rd coordinates of triangle:150
```



Enter translatio factors :

20

30



09. PROGRAM TO REFLECT A TRIANGLE

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<process.h>
#include<math.h>

void main()
{
clrscr();

int graphdriver=DETECT,graphmode;
initgraph(&graphdriver,&graphmode,"...\\bgi");

int x,y,x1,a[3][3];
double b[3][3],c[3][3];

cout<<"\n      Enter 1st coordinates of triangle:";
cin>>a[0][0]>>a[1][0];

cout<<"\n      Enter 2nd coordinates of triangle:";
cin>>a[0][1]>>a[1][1];

cout<<"\n      Enter 3rd coordinates of triangle:";
cin>>a[0][2]>>a[1][2];

cout<<"\n Enter 1. for reflection in x-axis:\n";
cout<<"\n Enter 2. for reflection in y-axis:\n";
cout<<"\n Enter 3. for reflection in both the axis:\n";
cin>>x;

cleardevice();
line(320,0,320,479);
line(0,240,639,240);
```

```
line(a[0][0],a[1][0],a[0][1],a[1][1]);
line(a[0][1],a[1][1],a[0][2],a[1][2]);
line(a[0][0],a[1][0],a[0][2],a[1][2]);
switch(x)
{
case 1:b[0][0]=640-a[0][0];
      b[0][1]=640-a[0][1];
      b[0][2]=640-a[0][2];
      b[1][0]=a[1][0];
      b[1][1]=a[1][1];
      b[1][2]=a[1][2];
      line(320,0,320,479);
      line(0,240,639,240);
      line(b[0][0],b[1][0],b[0][1],b[1][1]);
      line(b[0][1],b[1][1],b[0][2],b[1][2]);
      line(b[0][0],b[1][0],b[0][2],b[1][2]);
      getch();
      break;
case 2:b[1][0]=480-a[1][0];
      b[1][1]=480-a[1][1];
      b[1][2]=480-a[1][2];
      b[0][0]=a[0][0];
      b[0][1]=a[0][1];
      b[0][2]=a[0][2];
      line(320,0,320,479);
      line(0,240,639,240);
      line(b[0][0],b[1][0],b[0][1],b[1][1]);
      line(b[0][1],b[1][1],b[0][2],b[1][2]);
      line(b[0][0],b[1][0],b[0][2],b[1][2]);
      getch();
      break;
```

```

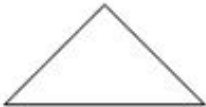
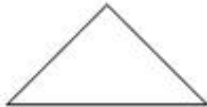

case 3: b[0][0]=640-a[0][0];
        b[0][1]=640-a[0][1];
        b[0][2]=640-a[0][2];
        b[1][0]=a[1][0];
        b[1][1]=a[1][1];
        b[1][2]=a[1][2];
        line(320,0,320,479);
        line(0,240,639,240);
        line(b[0][0],b[1][0],b[0][1],b[1][1]);
        line(b[0][1],b[1][1],b[0][2],b[1][2]);
        line(b[0][0],b[1][0],b[0][2],b[1][2]);
        b[1][0]=480-a[1][0];
        b[1][1]=480-a[1][1];
        b[1][2]=480-a[1][2];
        b[0][0]=a[0][0];
        b[0][1]=a[0][1];
        b[0][2]=a[0][2];
        line(320,0,320,479);
        line(0,240,639,240);
        line(b[0][0],b[1][0],b[0][1],b[1][1]);
        line(b[0][1],b[1][1],b[0][2],b[1][2]);
        line(b[0][0],b[1][0],b[0][2],b[1][2]);
        getch();
        break;
}
getch();
closegraph();

}

```

OUTPUT

```
Enter 1st coordinates of triangle:100
100
Enter 2nd coordinates of triangle:200
100
Enter 3rd coordinates of triangle:150
50
Enter 1. for reflection in x-axis:
Enter 2. for reflection in y-axis:
Enter 3. for reflection in both the axis:
3
```

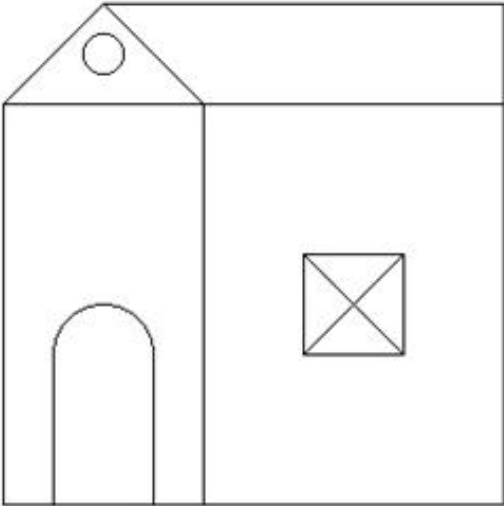
	
	

10. PROGRAM TO DRAW A HUT USING SIMPLE GRAPHIC FUNCTIONS

```
#include<conio.h>
#include<iostream.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
#include<process.h>
void main()
{
int graphdriver=DETECT,graphmode;
initgraph(&graphdriver,&graphmode,"...\\bgi");
line(100,100,150,50);
line(150,50,200,100);
line(100,100,200,100);
line(150,50,350,50);
line(200,100,350,100);
line(350,50,350,100);
circle(150,75,10);
rectangle(100,100,200,300);
rectangle(200,100,350,300);
rectangle(250,175,300,225);
line(250,175,300,225);
line(300,175,250,225);
line(125,300,125,225);
line(175,300,175,225);
arc(150,225,0,180,25);
getch();
closegraph();

}
```

OUTPUT



11.PROGRAM TO FILL A POLYGON

```
#include<conio.h>
#include<iostream.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
#include<process.h>
void main()
{
int graphdriver=DETECT,graphmode;
initgraph(&graphdriver,&graphmode,"...\\bgi"); int
p=1,x;

int    a[12]={ 100,100,150,150,200,100,200,200,100,200,100,100};
drawpoly(6,a);

for(int  i=100;i<200;i++)
    {   p=1;
for(int  j=100;j<=200;j++)
    {
    x=getpixel(j,i);
    for(int  d=0;d<11;d++)
    {
    if(j==a[d]&& i==a[d+1] )
        break;
    else
    {
        if(x>0&&d==10)
            p++;
    }
    if(p%2==0)
        putpixel(j,i,4);
    }
```

```
    }  
    }  
    }  
getch();  
closegraph();  
}
```

OUTPUT



12. To make an isometric dimensional drawing of a connecting rod using isometric grid and Snap.

The connecting rod in the connection b/w the piston and crank shaft. It joins the wrist pin of the piston with the throw or crank pin of the crankshaft. The lighter the connection rod and piston greater the resulting power and lesser the vibration because the reciprocation weight is less.

Isometric drawing are generally used to help visualize the shape of an object. It is much easier to conceive the shape of the object.

An Isometric drawing should not be confused with a three-dimensional (3D) drawing. An Isometric Drawing is just a two-dimensional representation of a three-dimensional drawing in 2D plane.

PROCEDURE:-

GRID:- This will make the whole screen into a graph paper

SNAP:- If you select snap or s, auto cad takes that you want the specification to grid spacing requirements.

Tools > Drafting Setting > Select SNAP AND GRID tab > SNAP TYPE & STYLE > Select Isometric snap > OK.

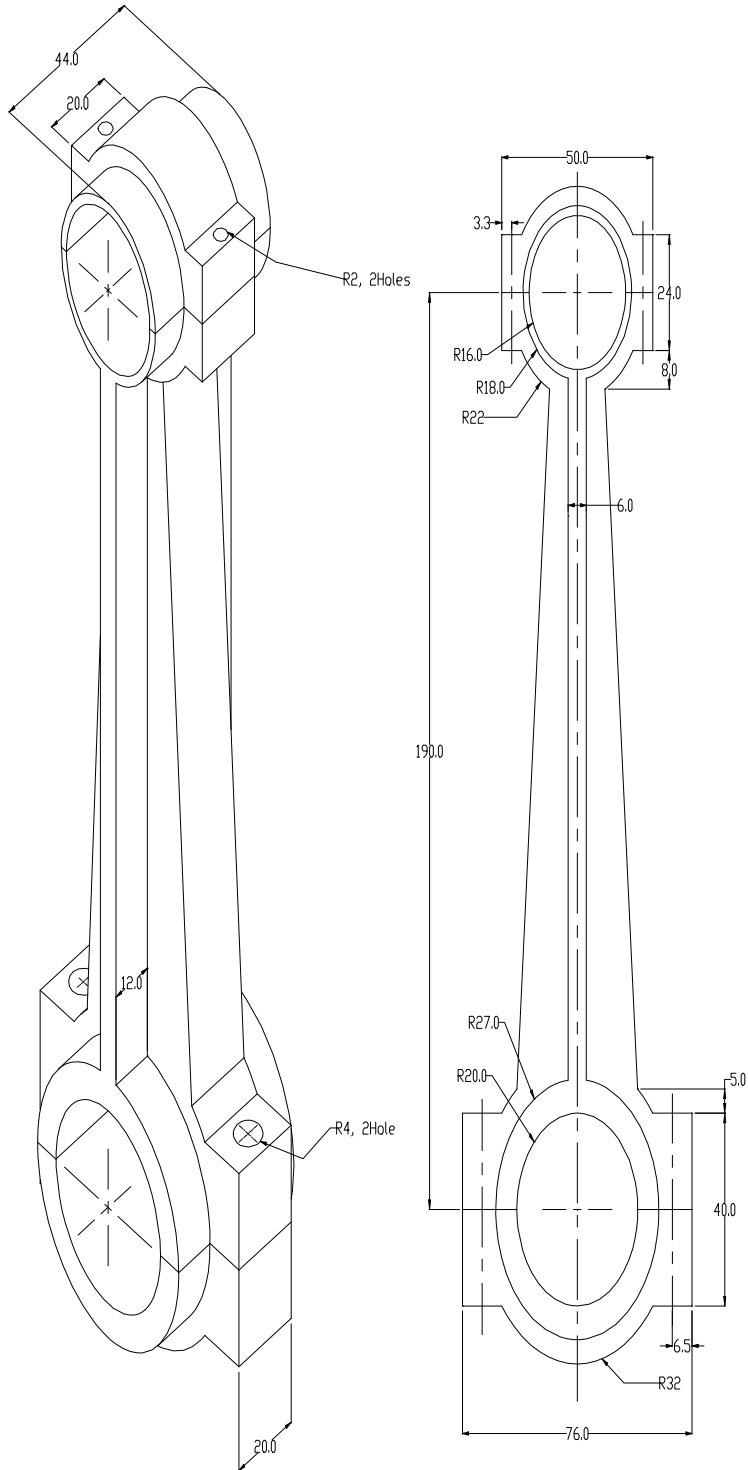
The crosshairs are displayed at an isometric angle, and their orientation depends on the current isoplane.

You can change the isoplane by pressing CTRL+E or F5.

Drawing Isometric Circle:-

Isometric circles are drawn by using the ELLIPS command and then selecting the Isocircle option. Before you enter the radius or diameter of the isometric circle, you must make sure you are in the required isoplane.

CONNECTING ROD
(Exp.-03)



13. Draw quarter sectional isometric view of a cotter joint.

Isometric drawings are generally used to help visualize the shape of an object. An isometric drawing is just a two-dimensional representation of a three-dimensional drawing in a 2D plane.

Procedure for setting the snap and grid:-

Command: snap

Specify snap spacing or [ON/OFF/Aspect/Rotate/Style/Type] <10.0000>: s

Enter snap grid style [Standard/Isometric] <S>: i

Command: _line

Specify first point:

Specify next point or [Undo]:

Command: ellipse

Specify axis endpoint of ellipse or [Arc/Center/Isocircle]: i

Specify center of isocircle:

Specify radius of isocircle or [Diameter]: 12

Command: copy

Select objects: <Snap off> 1 found

Select objects:

Specify base point or displacement, or [Multiple]: Specify second point of displacement or <use first point as displacement>: <Ortho on> <Isoplane Top> 30

Command: _line

Specify first point:

Specify next point or [Undo]:

Similarly draw the isocircles of radius 22, 15, 28 at their respective position as mentioned in drawing

Command: trim

Select objects:

Command: erase

Select objects:

Command: _arc

Specify start point of arc or [Center]:

Specify second point of arc or [Center/End]:

Specify end point of arc:

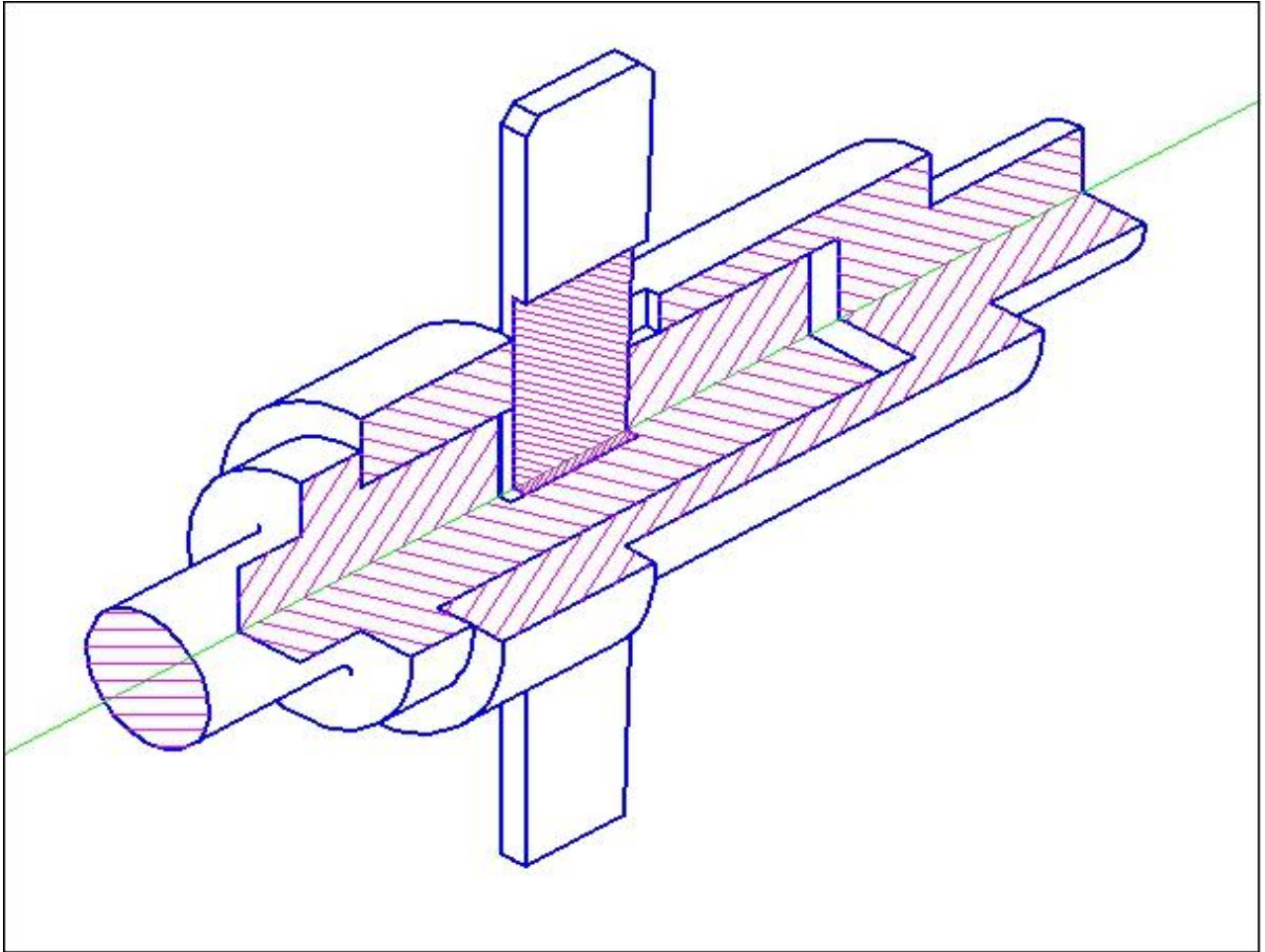
Command: hatch

Select internal point:

Specify angle

Command: save

Quarter Sectional Isometric View of a Cotter-Joint
(Exp.-04)



14. Draw Different type's bolts and nuts with internal and external threading in Acme and Square threading standards. Save the bolts and nut as blocks suitable for insertion.

Procedure:- List of commands for drawing bolts and nut.

- i) Polygon
- ii) Circle
- iii) Line
- iv) Line type (Center line, Hidden line etc.)
- v) Hatch
- vi) Block

- i) Polygon:- A regular polygon is a closed geometric figure with equal sides and equal angles. The number of sides varies from 3 to 1024.

Command: polygon

Command: Enter number of sides <4>: (Ex: 6)

Specify center of polygon or [Edge]: (Ex: 10, 10)

Enter an option [Inscribed in circle/Circumscribed about circle] <I>: I (I or C)

Specify radius of circle: (Ex: 50)

- ii) Circle:-

Command: circle

Command: _ Specify center point for circle or [3P/2P/Ttr (tan tan radius)]: (Ex: 10, 10)

Specify radius of circle or [Diameter]: (Ex: 50 or d)

- iii) Line:-

Command: line

Command: Specify first point: (Ex: 10, 10)

Specify next point or [Undo]: (Ex: 100, 100)

- iv) Linetype:-

Command: linetype

Format > Linetype (It will display a Linetype Manager dialog box) > Load > select the required linetype from the given linetype list > ok.

v) Hatch:-

Command: bhatch

The Hatch command allows you to hatch a region enclosed within a boundary (closed area) by selecting a point inside the boundary or by selection the objects to be hatched.

Draw > Hatch (It will display a Boundary Hatch and Fill dialog box) > specify Hatch type > Hatch pattern > Angle > Scale > Choose pick point (select any point inside the object) or Select object > Choose Preview or Ok.

vii) Block:-

Command: _block

Draw > Block > Make (it will display a Block Definition dialog box)

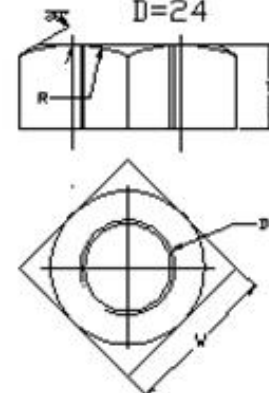
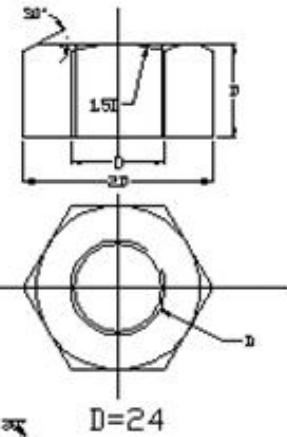
- e) Specify the Block Name.
- f) Specify the insertion Base Point (This point is used as a reference point to insert the block by choosing Pick Point)
- g) Select object.
- h) OK.

Command: Insert

Insert > Block > Select the Block Name from the drop down list > Ok.

Nut, Bolt & Thread
(Exn.-05)

Hexagonal Nut



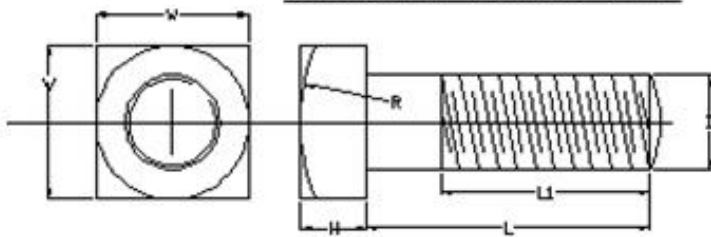
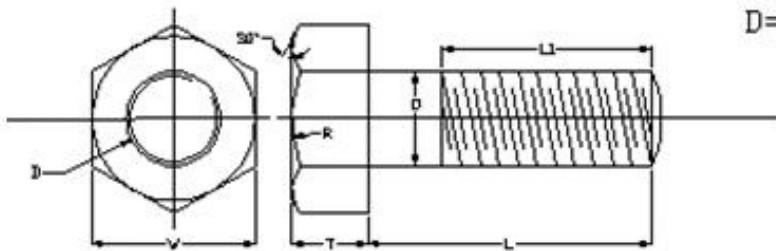
Square Nut

$W = 1.5D + 3$
 $T = 0.8D$
 $R = 1.5D$
 $L = 3D$
 $L_1 = 2D$
 $D = 24$

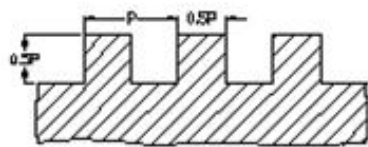
$W = 1.5D + 3$
 $H = 0.7D$
 $R = 2D$
 $L = 3D$
 $L_1 = 2D$
 $D = 24$

$T = 0.9D$
 $W = 1.5D + 3$
 $R = 1.5D$
 $D = 24$

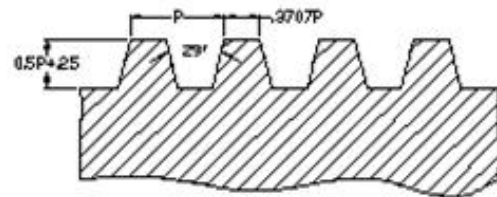
Hexagonal Headed bolt



Square Headed bolt



P=24
Square Threads



P=24
Acme Threads

15. Draw 3D models by extruding simple 2D objects, dimension and name the objects.

Procedure:- List of commands for 3D models.

- i) 3D View
- ii) Region
- iii) Extrude
- iv) Move
- v) Text
- vi) Dimension
- vii) Shade

i) 3D View

View > 3D view > Choose any one of the view (Front/ Top/ Left etc.)

ii) Region

Draw > Region > Select object.

iii) Extrude

Draw > Solids > Extrude > Select object > Specify height > Specify angle.

iv) Move

Modify > Move > Select object > Specify base point > Specify next point.

v) Text

Draw > Text > Single Line Text > Specify Start Point > Specify height > Specify rotation angle > Enter Text.

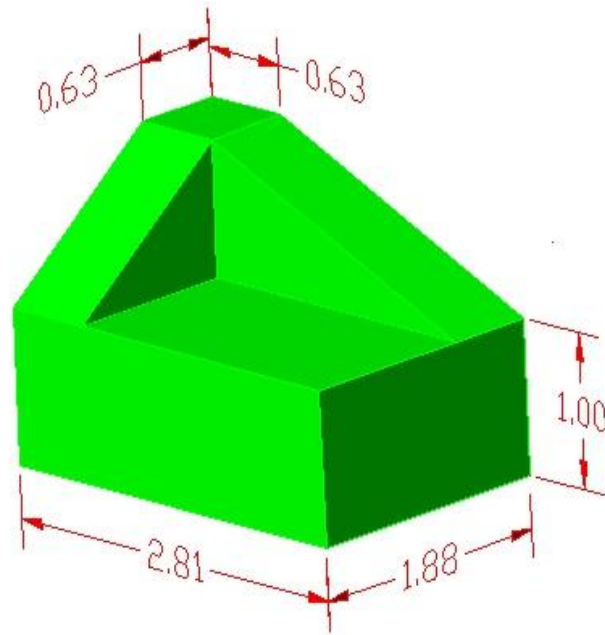
vi) Dimension

Dimension > Linear > Specify First Extension Line Origin > Specify Second Extension Line Origin.

vii) Shade

View > Shade > Select the required shade (Ex.- 2d wire frame/ Hidden/ Flat shaded/ Gouraud shaded, edges on etc.).

3D Model
(Exp.-06)



16. Draw a Spiral by extruding a circle.

Procedure:- List of commands for Draw the Spiral.

- i) Helix
- ii) Free Orbit
- iii) Circle
- iv) Sweep
- v) Visual Styles

i) Helix

Draw > Helix.

Command: Helix

Specify center point of base: 0,0

Specify base radius or [Diameter] <1.0000>: 10

Specify top radius or [Diameter] <10.0000>: 10

Specify helix height or [Axis endpoint/Turns/turn Height/tWist] <1.0000>: 50

ii) Free Orbit

View > Orbit > Free Orbit

iii) Circle

Draw > Circle > Center, Radius

Command: circle

Command: _ Specify center point for circle or [3P/2P/Ttr (tan tan radius)]: (Select any one end of the helix)

Command: Specify radius of circle or [Diameter]: (Ex: 50 or d)

iv) Sweep

Draw > Modeling > Sweep

Command: _sweep

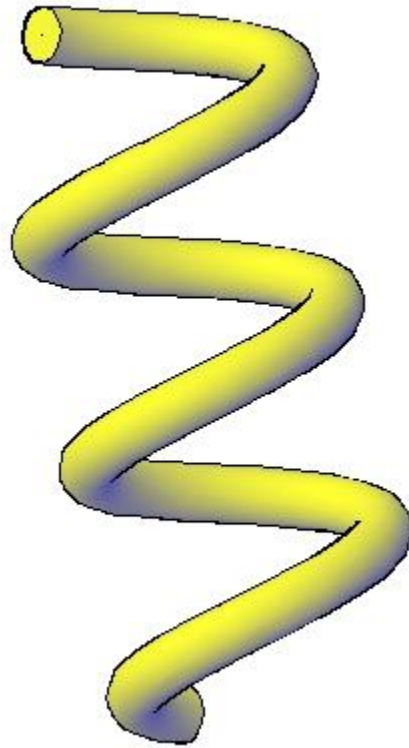
Select objects to sweep: (Select the circle)

Select sweep path or [Alignment/Base point/Scale/Twist]: (Select the helix)

v) Visual Styles

View > Visual Styles > Conceptual

Spiral
(Exp.-07)



17. CNC PROGRAM FOR FACING OPERATION

Program:
[BILLET X25 Z90;
G21;
G98;
G28 U0 W0;
G50 S1000;
G00 X26 Z1;
G00 X26 Z-3;
G00 X0 Z-3 F5;
G00 X0 Z1;
G00 X26 Z1;
G28 U0 W0;
M05;
M30;
:

18. CNC PROGRAM FOR TURNING OPERATION

Program:
[BILLET X25 Z90;
G21;
G98;
G28 U0 W0;
G50 S1000;
G00 X26 Z1;
G00 X26 Z-3;
G00 X0 Z-3 F5;
G00 X0 Z1;
G00 X26 Z1;
G28 U0 W0;
M05;
M30;

19 .CNC PROGRAM FOR BOX TURNING & FACING OPERATION

Program:
[BILLET X25 Z90;
G21;
G98;

```
G28 U0 W0;
G50 S1000;
G00 X15 Z1;
G94 X0 Z-1 F5;
Z-2;
Z-3;
Z-4;
Z-5;
Z-6;
Z-7;
Z-8;
Z-9;
Z-10;
G90 X15 Z-50 F5;
X14 ;
X13 ;
X12 ;
X11 ;
X10 ;
G90 X10 Z-30 F5 ;
X9 ;
X8 ;
G28 U0 W0;
M05;
M30;
```

19.CNC PROGRAM FOR TAPER TURNING OPERATION

Program:

```
[BILLET X25 Z90;
G21;
G98;
G28 U0 W0;
G50 S1000;
G00 X25 Z0;
G90 X23 Z-39 F25;
X20 Z-34;
X18 Z-10;
X16 Z-10;
X14 Z-10;
X12 Z-10;
X10 Z-10 ;
G90 X10 Z-10 R-1 F25 ;
X10 R-2 ;
X10 R-3 ;
X10 R-4 ;
```

X10 R-5 ;
G00 X20 Z-15 ;
G00 X19 Z-22 R0.5 F25 ;
X18 R1 ;
X17 R1.5 ;
X16 R2 ;
X15 R2.5;
G00 X20 Z-22;
G90 X20 Z-29 R0.5 F25;
X20 R-1;
X20 R-1.5;
X20 R-2;
X20 R-2.5;
G28 U0 W0;
M05;
M30;