Line_1_Direct_Application

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;

int main()
{
   int gd = DETECT, gm;
   int n;
   initgraph(&gd, &gm, "");

   // declare two points
   int x1, y1, x2, y2;
   cout<< "Enter x1 = ";
   cin>> x1;
   cout<< "Enter y1 = ";
   cin>> y1;
   cout<< "Enter x2 = ";
   cin>> x2;
   cout<< "Enter y2 = ";
   cin>> y2;

   int dx, dy, m, b;

   dx = x2 - x1;
   dy = y2 - y1;
   m = dy/dx;
   b = y1 - m * x1;
```



```
   int x, y, xend, color = 15;

   if(dx < 0)
   {
      x = x2;
      y = y2;
      xend = x1;
   } else
   {
      x = x1;
      x = y1;
      xend = x2;
   }

   putpixel(x, y, color);
   while(x != xend)
   {
      x += 1;
      y = m * x + b;
      putpixel(x, y, color);
   }

   cin>> n;
   return 0;
}
```

Line_2_DDA

```
#include<bits/stdc++.h>
#include<graphics.h>
```



```
using namespace std;

int main()
{
   int gd = DETECT, gm;
   int n;
   initgraph(&gd, &gm, "");

   // declare two points
   int x1, y1, x2, y2, color = 15;
   cout<< "Enter x1 = ";
   cin>> x1;
   cout<< "Enter y1 = ";
   cin>> y1;
   cout<< "Enter x2 = ";
   cin>> x2;
   cout<< "Enter y2 = ";
   cin>> y2;

   float dx, dy, step;

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

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



```
   }

   float x, y;
   float xinc, yinc;

   x = x1;
   y = y1;
   xinc = dx/step;
   yinc = dy/step;

   putpixel(x, y, color);
   for(int i = 1; i < step; i++)
   {
      x += xinc;
      y += yinc;
      putpixel(round(x), round(y), color);
      cout<< "\n" << round(x) << " " << round(y);
   }
   cin>> n;
   return 0;
}
```

Line_3_Bresenham

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;

int main()
{
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");
```

```
int x1, y1, x2, y2, color = 15;
cout<< "Enter x1 = ";
cin>> x1;
cout<< "Enter y1 = ";
cin>> y1;
cout<< "Enter x2 = ";
cin>> x2;
cout<< "Enter y2 = ";
cin>> y2;

int dx, dy,  p, x, y;
dx = x2 - x1;
dy = y2 -y1;
p = 2 * dy - dx;
x = x1;
y = y1;

for(int i = 0; i <=dx; i++)
{
   if(p < 0)
   {
      putpixel(x, y, color);
      p += 2*dy;
      x += 1;
   } else
   {
      putpixel(x, y, color);
      p += 2*dy - 2*dx;
      x += 1;
      y += 1;
```



```
   }
}

getch();
return 0;
}
```

Line_1_Direct_Application

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;

int main()
{
   int gd = DETECT, gm;
   int n;
   initgraph(&gd, &gm, "");

   // declare two points
   int x1, y1, x2, y2;
   cout<< "Enter x1 = ";
   cin>> x1;
   cout<< "Enter y1 = ";
   cin>> y1;
   cout<< "Enter x2 = ";
   cin>> x2;
   cout<< "Enter y2 = ";
   cin>> y2;

   int dx, dy, m, b;

   dx = x2 - x1;
```



```
   dy = y2 - y1;
   m = dy/dx;
   b = y1 - m * x1;

   int x, y, xend, color = 15;

   if(dx < 0)
   {
      x = x2;
      y = y2;
      xend = x1;
   } else
   {
      x = x1;
      x = y1;
      xend = x2;
   }

   putpixel(x, y, color);
   while(x != xend)
   {
      x += 1;
      y = m * x + b;
      putpixel(x, y, color);
   }

   cin>> n;
   return 0;
}
```

Circle_using_equation


```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;

int main()
{
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");

   int r, x = 0, y, color = 15;
   cout<< "Enter Radius = ";
   cin>> r;

   // for displaying circle in positive portion
   int xc = 110, yc = 130;

   y = sqrt((r*r - x*x));
   while(x < r/sqrt(2))
   {
      putpixel(xc + x, yc + y, color);
      putpixel(xc - x, yc+ y, color);
      putpixel(xc + x, yc - y, color);
      putpixel(xc - x, yc- y, color);
      putpixel(xc + y, yc + x, color);
      putpixel(xc - y, yc + x, color);
      putpixel(xc + y, yc - x, color);
      putpixel(xc - y, yc - x, color);

      x += 1;
```

```
    y = sqrt((r*r - x*x));
  }

  getch();
  return 0;
}
```

Circle_MidPoint

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;

void plotCircle(int x, int y)
{
  // for displaying circle in positive portion
  int xc = 110, yc = 130, color = 15;

  putpixel(xc + x, yc + y, color);
  putpixel(xc - x, yc+ y, color);
  putpixel(xc + x, yc - y, color);
  putpixel(xc - x, yc- y, color);
  putpixel(xc + y, yc + x, color);
  putpixel(xc - y, yc + x, color);
  putpixel(xc + y, yc - x, color);
  putpixel(xc - y, yc - x, color);

}
int main()
{
  int gd = DETECT, gm;
  initgraph(&gd, &gm, "");
```



```
  int  r,  color = 15;
  cout<< "Enter Radius = ";
  cin>> r;

  int x = 0, y = r, p = 1 - r;

  plotCircle(x, y);
  while( x <= y)
  {
    if(p < 0)
    {
      p = p + 2 * x + 3;
    }
    else
    {
      p = p + 2 * (x - y) + 5;
      y--;
    }
    x++;

    plotCircle(x, y);
  }

  getch();
  return 0;
}
```

Circle_Bresenham


```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;

void plotCircle(int x, int y)
{
  // for displaying circle in positive portion
  int xc = 110, yc = 130, color = 15;

  putpixel(xc + x, yc + y, color);
  putpixel(xc - x, yc+ y, color);
  putpixel(xc + x, yc - y, color);
  putpixel(xc - x, yc- y, color);
  putpixel(xc + y, yc + x, color);
  putpixel(xc - y, yc + x, color);
  putpixel(xc + y, yc - x, color);
  putpixel(xc - y, yc - x, color);

}
int main()
{
  int gd = DETECT, gm;
  initgraph(&gd, &gm, "");

  int  r,  color = 15;
  cout<< "Enter Radius = ";
  cin>> r;

  int x = 0, y = r, d = 3 - 2 * r;

  plotCircle(x, y);
```



```
  while( x <= y)
  {
    if(d < 0)
    {
      d = d + 4 * x + 6;
    }
    else
    {
      d = d + 4 * (x - y) + 10;
      y--;
    }
    x++;

    plotCircle(x, y);
  }

  getch();
  return 0;
}
```

Object_translation

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;

int main()
{
  int gd = DETECT, gm;
  initgraph(&gd, &gm, "");

  int x, y, r, tx, ty;
  cout<< "Enter value of Circle: \n\n";
```

```
    cout<< "x = ";
    cin>> x;
    cout<< "y = ";
    cin>> y;
    cout<< "radius: ";
    cin>> r;

    // initial circle
    circle(x, y, r);

    cout<< "Enter translation value of x: ";
    cin>> tx;
    cout<< "Enter translation value of y: ";
    cin>> ty;

    int newX, newY;

    // new circle after translation
    for(int i = 0, j = 0; i <= tx, j <= ty; i++, j++)
    {
        newX = x + i;
        newY = y + j;
        circle(newX, newY, r);
        delay(50);
        cleardevice();
    }
    circle(newX, newY, r);

    getch();
    return 0;
}
```


Object_rotation

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;

int main()
{
    int gd = DETECT, gm;
    int n;
    initgraph(&gd, &gm, "");

    int x1, y1, x2, y2, x3, y3, theta;
    cout<< "Enter value of Line: \n\n";
    cout<< "x1 = ";
    cin>> x1;
    cout<< "y1 = ";
    cin>> y1;
    cout<< "x2 = ";
    cin>> x2;
    cout<< "y2 = ";
    cin>> y2;

    // initial Line
    line(x1, y1, x2, y2);

    cout<< "Rotation Angle: ";
    cin>> theta;


    // calculation
    x3 = x1 * cos(theta) - y1 * sin(theta);
```



```
    y3 = y1 * cos(theta) + x1 * sin(theta);

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

    getch();
    return 0;
}
```

Object_scaling

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;

int main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");

    int x, y, r, sx, sy;
    cout<< "Enter value of Circle: \n\n";
    cout<< "x = ";
    cin>> x;
    cout<< "y = ";
    cin>> y;
    cout<< "radius: ";
    cin>> r;

    // initial circle
    circle(x, y, r);

    cout<< "Enter Scaling value of x: ";
```



```
    cin>> sx;
    cout<< "Enter Scaling value of y: ";
    cin>> sy;

    int newX, newY, newR;

    // new circle after Scaling
    for(int i = 0, j = 0; i <= sx, j <= sy; i++, j++)
    {
        newX = x * i;
        newY = y * j;
        newR = sqrt((newX*newX + newY * newY));
        circle(newX, newY, newR);
        delay(50);
        cleardevice();
    }
    circle(newX, newY, newR);

    getch();
    return 0;
}
```

Object_shearing

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;

int main()
{
    int gd = DETECT, gm;
    int n;
    initgraph(&gd, &gm, "");
```

```
int choice, x1=20, y1=20, x2=200, y2=200, shx, shy, g;

// initial rectangle
line(x1, y1, x1, y2);
line(x1, y2, x2, y2);
line(x2, y2, x2, y1);
line(x2, y1, x1, y1);

cout<< "1. X-Shear\n2. Y-Shear\n\nEnter your choice: ";
cin>> choice;


if(choice == 1)
{
   cout<< "Enter value of shx: ";
   cin>> shx;
   int newX1= x1 + y1*shx;

   cleardevice();

   // final quad after x-shear
   line(newX1, y1, x1, y2);
   line(x1, y2, x2, y2);
   line(x2, y2, newX1+x2, y1);
   line(newX1, y1, newX1+x2, y1);
}
else if(choice == 2)
{
   cout<< "Enter value of shy: ";
   cin>> shy;
```



```
   int newY1= y1 + x1*shy;

   cout<< newY1;
   cleardevice();

   // final quad after x-shear
   line(x1, y1, x1, y2);
   line(x1, y2, x2, y2+newY1);
   line(x2, y2+newY1, x2, newY1);
   line(x1, y1, x2, newY1);
}

getch();
return 0;
}
```

Object_reflection

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;

int main()
{
   int gd = DETECT, gm;
   int n;
   initgraph(&gd, &gm, "");

   int choice, x1=400, y1=50, x2=450, y2=100;

   /// Y-axis
   line(320, 0, 320, 480);
   /// X-axis
```



```
   line(0, 245, 640, 245);
   // initial rectangle
   line(x1, y1, x1, y2);
   line(x1, y2, x2, y2);
   line(x2, y2, x2, y1);
   line(x2, y1, x1, y1);

   // final rectangle after reflection
   line(x1, 480-y1, x1, 480-y2);
   line(x1, 480-y1, x2, 480-y1);
   line(x2, 480-y1, x2, 480-y2);
   line(x2, 480-y2, x1, 480-y2);

   getch();
   return 0;
}
```