#include<graphics.h>

#include<conio.h>

int main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

getch();

closegraph();

return 0;

}

#include <graphics.h>

int main()

{

int gd = DETECT, gm;

int x = 320, y = 240, radius;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

for ( radius = 25; radius <= 125 ; radius = radius + 20)

circle(x, y, radius);

getch();

closegraph();

return 0;

}

graphics program moving car

#include <graphics.h>

#include <dos.h>

int main()

{

int i, j = 0, gd = DETECT, gm;

initgraph(&gd,&gm,"C:**\\**TC**\\**BGI");

settextstyle(DEFAULT\_FONT,HORIZ\_DIR,2);

outtextxy(25,240,"Press any key to view the moving car");

getch();

for( i = 0 ; i <= 420 ; i = i + 10, j++ )

{

rectangle(50+i,275,150+i,400);

rectangle(150+i,350,200+i,400);

circle(75+i,410,10);

circle(175+i,410,10);

setcolor(j);

delay(100);

if( i == 420 )

**break**;

if ( j == 15 )

j = 2;

cleardevice(); *// clear screen*

}

getch();

closegraph();

return 0;

}

Drawing concentric circles

#include <graphics.h>

int main()

{

int gd = DETECT, gm;

int x = 320, y = 240, radius;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

for ( radius = 25; radius <= 125 ; radius = radius + 20)

circle(x, y, radius);

getch();

closegraph();

return 0;

}

Declaration :- void arc(int x, int y, int stangle, int endangle, int radius);  
arc function is used to draw an arc with center (x,y) and stangle specifies starting angle, endangle specifies the end angle and last parameter specifies the radius of the arc. arc function can also be used to draw a circle but for that starting angle and end angle should be 0 and 360 respectively.

**C programming source code for arc**

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

arc(100, 100, 0, 135, 50);

getch();

closegraph();

return 0;

}

Declaration :- void bar(int left, int top, int right, int bottom);

Bar function is used to draw a 2-dimensional, rectangular filled in bar . Coordinates of left top and right bottom corner are required to draw the bar. Left specifies the X-coordinate of top left corner, top specifies the Y-coordinate of top left corner, right specifies the X-coordinate of right bottom corner, bottom specifies the Y-coordinate of right bottom corner. Current fill pattern and fill color is used to fill the bar. To change fill pattern and fill color use [setfillstyle](http://www.programmingsimplified.com/c/graphics.h/setfillstyle).

## C programming code

#include <graphics.h>

#include <conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

bar(100, 100, 200, 200);

getch();

closegraph();

return 0;

}

Declaration :- void bar3d(int left, int top, int right, int bottom, int depth, int topflag);

bar3d function is used to draw a 2-dimensional, rectangular filled in bar . Coordinates of left top and right bottom corner of bar are required to draw the bar. left specifies the X-coordinate of top left corner, top specifies the Y-coordinate of top left corner, right specifies the X-coordinate of right bottom corner, bottom specifies the Y-coordinate of right bottom corner, depth specifies the depth of bar in pixels, topflag determines whether a 3 dimensional top is put on the bar or not ( if it is non-zero then it is put otherwise not ). Current fill pattern and fill color is used to fill the bar. To change fill pattern and fill color use setfillstyle.

### C program of bar3d

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

bar3d(100, 100, 200, 200, 20, 1);

getch();

closegraph();

return 0;

}

Declaration :- void circle(int x, int y, int radius);

Circle function is used to draw a circle with center (x,y) and third parameter specifies the radius of the circle. The code given below draws a circle.

### C program for circle

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

circle(100, 100, 50);

getch();

closegraph();

return 0;

}

In the above program (100, 100) are coordinates of center of the circle and 50 is the radius of circle.

Declaration :- void cleardevice();  
cleardevice function clears the screen in graphics mode and sets the current position to (0,0). Clearing the screen consists of filling the screen with current background color.

## C program for cleardevice

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

outtext("Press any key to clear the screen.");

getch();

cleardevice();

outtext("Press any key to exit...");

getch();

closegraph();

return 0;

}

Declaration :- void cleardevice();  
cleardevice function clears the screen in graphics mode and sets the current position to (0,0). Clearing the screen consists of filling the screen with current background color.

## C program for cleardevice

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

outtext("Press any key to clear the screen.");

getch();

cleardevice();

outtext("Press any key to exit...");

getch();

closegraph();

return 0;

}

Drawpoly function is used to draw polygons i.e. triangle, [rectangle](http://www.programmingsimplified.com/c/graphics.h/rectangle), pentagon, hexagon etc.

Declaration :- void drawpoly( int num, int \*polypoints );

num indicates (n+1) number of points where n is the number of vertices in a polygon, polypoints points to a sequence of (n\*2) integers . Each pair of integers gives x and y coordinates of a point on the polygon. We specify (n+1) points as first point coordinates should be equal to (n+1)th to draw a complete figure.

To understand more clearly we will draw a triangle using drawpoly, consider for example the array :-  
int points[] = { 320, 150, 420, 300, 250, 300, 320, 150};

points array contains coordinates of triangle which are (320, 150), (420, 300) and (250, 300). Note that last point(320, 150) in array is same as first. See the program below and then its output, it will further clear your understanding.

## C program for drawpoly

#include <graphics.h>

#include <conio.h>

main()

{

int gd=DETECT,gm,points[]={320,150,420,300,250,300,320,150};

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

drawpoly(4, points);

getch();

closegraph();

return 0;

}

Declaration of fillellipse function :-  
void fillellipse(int x, int y, int xradius, int yradius);  
x and y are coordinates of center of the ellipse, xradius and yradius are x and y radius of ellipse respectively.

### C program for fillellipse

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

fillellipse(100, 100, 50, 25);

getch();

closegraph();

return 0;

}

Fillpoly function draws and fills a polygon. It require same arguments as [drawpoly](http://www.programmingsimplified.com/c/graphics.h/drawpoly).

Declaration :- void drawpoly( int num, int \*polypoints );  
For details of arguments see [drawpoly](http://www.programmingsimplified.com/c/graphics.h/drawpoly).  
fillpoly fills using current fill pattern and color which can be changed using [setfillstyle](http://www.programmingsimplified.com/c/graphics.h/setfillstyle).

## C programming code

#include <graphics.h>

#include <conio.h>

main()

{

int gd=DETECT,gm,points[]={320,150,440,340,230,340,320,150};

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

fillpoly(4, points);

getch();

closegraph();

return 0;

}

Declaration :- void floodfill(int x, int y, int border);

floodfill function is used to fill an enclosed area. Current fill pattern and fill color is used to fill the area.(x, y) is any point on the screen if (x,y) lies inside the area then inside will be filled otherwise outside will be filled,border specifies the color of boundary of area. To change fill pattern and fill color use setfillstyle. Code given below draws a circle and then fills it.

## C programming code

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

setcolor(RED);

circle(100,100,50);

floodfill(100,100,RED);

getch();

closegraph();

return 0;

}

In the above program a circle is drawn in RED color. Point (100,100) lies inside the circle as it is the center of circle, third argument to floodfill is RED which is color of boundary of circle. So the output of above program will be a circle filled with WHITE color as it is the default fill color.

Declaration :- void getarccoords(struct arccoordstype \*var);

getarccoords function is used to get coordinates of arc which is drawn most recently. arccoordstype is a predefined structure which is defined as follows:

struct arccoordstype

{

int x, y; */\* center point of arc \*/*

int xstart, ystart; */\* start position \*/*

int xend, yend; */\* end position \*/*

};

address of a structure variable of type arccoordstype is passed to function getarccoords.

## C program of getarccoords

#include<graphics.h>

#include<conio.h>

#include<stdio.h>

main()

{

int gd = DETECT, gm;

struct arccoordstype a;

char arr[100];

initgraph(&gd, &gm,"C:**\\**TC**\\**BGI");

arc(250,200,0,90,100);

getarccoords(&a);

sprintf(arr,"(%d, %d)",a.xstart,a.ystart);

outtextxy(360,195,arr);

sprintf(arr,"(%d, %d)",a.xend,a.yend);

outtextxy(245,85,arr);

getch();

closegraph();

return 0;

}

In the above program we have drawn an arc and then we get the coordinates of end points of arc using getarccoords.Coordinates so obtained are displayed using

Declarations of ellipse function :-  
void ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius);

Ellipse is used to draw an ellipse (x,y) are coordinates of center of the ellipse, stangle is the starting angle, end angle is the ending angle, and fifth and sixth parameters specifies the X and Y radius of the ellipse. To draw a complete ellipse strangles and end angle should be 0 and 360 respectively.

## C programming code for ellipse

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

ellipse(100, 100, 0, 360, 50, 25);

getch();

closegraph();

return 0;

}

getbkcolor function returns the current background color

Declaration : int getbkcolor();

e.g. color = getbkcolor(); // color is an int variable  
if current background color is GREEN then color will be 2.

## C program for getbkcolor

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm, bkcolor;

char a[100];

initgraph(&gd,&gm,"C:**\\**TC**\\**BGI");

bkcolor = getbkcolor();

sprintf(a,"Current background color = %d", bkcolor);

outtextxy( 10, 10, a);

getch();

closegraph();

return 0;

}

getcolor function returns the current drawing color.

Declaration : int getcolor();

e.g. a = getcolor(); // a is an integer variable  
if current drawing color is WHITE then a will be 15.  
See [colors in c graphics](http://www.programmingsimplified.com/c/graphics.h/colors).

## C programming code for getcolor

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm, drawing\_color;

char a[100];

initgraph(&gd,&gm,"C:**\\**TC**\\**BGI");

drawing\_color = getcolor();

sprintf(a,"Current drawing color = %d", drawing\_color);

outtextxy( 10, 10, a );

getch();

closegraph();

return 0;

}

getdrivername function returns a pointer to the current graphics driver.

## C program for getdrivername

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

char \*drivername;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

drivername = getdrivername();

outtextxy(200, 200, drivername);

getch();

closegraph();

return 0;

}

getmaxcolor function returns maximum color value for current graphics mode and driver. Total number of colors available for current graphics mode and driver are ( getmaxcolor() + 1 ) as color numbering starts from zero.

Declaration :- int getmaxcolor();

## C program of getmaxcolor

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm, max\_colors;

char a[100];

initgraph(&gd,&gm,"C:**\\**TC**\\**BGI");

max\_colors = getmaxcolor();

sprintf(a,"Maximum number of colors for current graphics mode and driver = %d",max\_colors+1);

outtextxy(0, 40, a);

getch();

closegraph();

return 0;

}

This animation using c draws a smiling face which appears at random position on screen. See output below the code, it will help you in understanding the code easily.

## C programming code

#include<graphics.h>

#include<conio.h>

#include<stdlib.h>

main()

{

int gd = DETECT, gm, area, temp1, temp2, left = 25, top = 75;

void \*p;

initgraph(&gd,&gm,"C:**\\**TC**\\**BGI");

setcolor(YELLOW);

circle(50,100,25);

setfillstyle(SOLID\_FILL,YELLOW);

floodfill(50,100,YELLOW);

setcolor(BLACK);

setfillstyle(SOLID\_FILL,BLACK);

fillellipse(44,85,2,6);

fillellipse(56,85,2,6);

ellipse(50,100,205,335,20,9);

ellipse(50,100,205,335,20,10);

ellipse(50,100,205,335,20,11);

area = imagesize(left, top, left + 50, top + 50);

p = malloc(area);

setcolor(WHITE);

settextstyle(SANS\_SERIF\_FONT,HORIZ\_DIR,2);

outtextxy(155,451,"Smiling Face Animation");

setcolor(BLUE);

rectangle(0,0,639,449);

while(!kbhit())

{

temp1 = 1 + random ( 588 );

temp2 = 1 + random ( 380 );

getimage(left, top, left + 50, top + 50, p);

putimage(left, top, p, XOR\_PUT);

putimage(temp1 , temp2, p, XOR\_PUT);

delay(100);

left = temp1;

top = temp2;

}

getch();

closegraph();

return 0;

}

This animation using c draws a smiling face which appears at random position on screen. See output below the code, it will help you in understanding the code easily.

## C programming code

#include<graphics.h>

#include<conio.h>

#include<stdlib.h>

main()

{

int gd = DETECT, gm, area, temp1, temp2, left = 25, top = 75;

void \*p;

initgraph(&gd,&gm,"C:**\\**TC**\\**BGI");

setcolor(YELLOW);

circle(50,100,25);

setfillstyle(SOLID\_FILL,YELLOW);

floodfill(50,100,YELLOW);

setcolor(BLACK);

setfillstyle(SOLID\_FILL,BLACK);

fillellipse(44,85,2,6);

fillellipse(56,85,2,6);

ellipse(50,100,205,335,20,9);

ellipse(50,100,205,335,20,10);

ellipse(50,100,205,335,20,11);

area = imagesize(left, top, left + 50, top + 50);

p = malloc(area);

setcolor(WHITE);

settextstyle(SANS\_SERIF\_FONT,HORIZ\_DIR,2);

outtextxy(155,451,"Smiling Face Animation");

setcolor(BLUE);

rectangle(0,0,639,449);

while(!kbhit())

{

temp1 = 1 + random ( 588 );

temp2 = 1 + random ( 380 );

getimage(left, top, left + 50, top + 50, p);

putimage(left, top, p, XOR\_PUT);

putimage(temp1 , temp2, p, XOR\_PUT);

delay(100);

left = temp1;

top = temp2;

}

getch();

closegraph();

return 0;

}

getx function returns the X coordinate of current position.

Declaration :- int getx();

## C program of getx

#include <graphics.h>

#include <conio.h>

main()

{

int gd = DETECT, gm;

char array[100];

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

sprintf(array, "Current position of x = %d",getx());

outtext(array);

getch();

closegraph();

return 0;

}

gety function returns the y coordinate of current position.

Declaration :- int gety();

## C programming source code for gety

#include <graphics.h>

#include <conio.h>

main()

{

int gd = DETECT, gm, y;

char array[100];

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

y = gety();

sprintf(array, "Current position of y = %d", y);

outtext(array);

getch();

closegraph();

return 0;

}

graphdefaults function resets all graphics settings to their defaults.

Declaration :- void graphdefaults();

It resets the following graphics settings :-

* Sets the viewport to the entire screen.
* Moves the current position to (0,0).
* Sets the default palette colors, background color, and drawing color.
* Sets the default fill style and pattern.
* Sets the default text font and justification.

**C programming source code for graphdefaults**

#include <graphics.h>

#include <conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

setcolor(RED);

setbkcolor(YELLOW);

circle(250, 250, 50);

getch();

graphdefaults();

getch();

closegraph();

return 0;

}

In the above program we have first changed the drawing color to RED and background color to YELLOW and then drawn a circle with (250, 250) as center and 50 as radius. When the user will press a key graphdefaults is called and both drawing and background color will be reset to their default values i.e. WHITE and BLACK respectively.

grapherrormsg function returns an error message string.

Declaration :- char \*grapherrormsg( int errorcode );

## C programming code for grapherrormsg

#include <graphics.h>

#include <stdlib.h>

#include <conio.h>

main()

{

int gd, gm, errorcode;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

errorcode = graphresult();

if(errorcode != grOk)

{

printf("Graphics error: %s**\n**", grapherrormsg(errorcode));

printf("Press any key to exit.");

getch();

exit(1);

}

getch();

closegraph();

return 0;

}

In the above program we have not written gd = DETECT.

imagesize function returns the number of bytes required to store a bitimage. This function is used when we are using [getimage](http://www.programmingsimplified.com/c/graphics.h/getimage).

Declaration:- unsigned int imagesize(int left, int top, int right, int bottom);

## C programming code for imagesize

#include <graphics.h>

#include <conio.h>

main()

{

int gd = DETECT, gm, bytes;

char array[100];

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

circle(200, 200, 50);

line(150, 200, 250, 200);

line(200, 150, 200, 250);

bytes = imagesize(150, 150, 250, 250);

sprintf(array, "Number of bytes required to store required area = %d", bytes);

outtextxy(10, 280, array);

getch();

closegraph();

return 0;

}

line function is used to draw a line from a point(x1,y1) to point(x2,y2) i.e. (x1,y1) and (x2,y2) are end points of the line.The code given below draws a line.

Declaration :- void line(int x1, int y1, int x2, int y2);

## C programming code for line

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

line(100, 100, 200, 200);

getch();

closegraph();

return 0;

}

ineto function draws a line from current position(CP) to the point(x,y), you can get current position using getx and gety function.

### C programming code for lineto

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

moveto(100, 100);

lineto(200, 200);

getch();

closegraph();

return 0;

}

Linerel function draws a line from the current position(CP) to a point that is a relative distance (x, y) from the CP, then advances the CP by (x, y). You can use [getx](http://www.programmingsimplified.com/c/graphics.h/getx) and [gety](http://www.programmingsimplified.com/c/graphics.h/gety) to find the current position.

## C programming code for linerel

#include <graphics.h>

#include <conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

moveto(250, 250);

linerel(100, -100);

getch();

closegraph();

return 0;

}

moveto function changes the current position (CP) to (x, y)

Declaration :- void moveto(int x, int y);

## C programming code for moveto

#include <graphics.h>

#include <conio.h>

main()

{

int gd = DETECT, gm;

char msg[100];

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

sprintf(msg, "X = %d, Y = %d",getx(),gety());

outtext(msg);

moveto(50, 50);

sprintf(msg, "X = %d, Y = %d", getx(), gety());

outtext(msg);

getch();

closegraph();

return 0;

}

moverel function moves the current position to a relative distance.

Declaration :- void moverel(int x, int y);

## C programming code for moverel

#include <graphics.h>

#include <conio.h>

main()

{

int gd = DETECT, gm, x, y;

char message[100];

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

moveto(100, 100);

moverel(100, -100);

x = getx();

y = gety();

sprintf(message, "Current x position = %d and y position = %d", x, y);

outtextxy(10, 10, message);

getch();

closegraph();

return 0;

}

outtext function displays text at current position.

Declaration :- void outtext(char \*string);

## C programming code for outtext

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

outtext("To display text at a particular position on the screen use outtextxy");

getch();

closegraph();

return 0;

}

Do not use text mode functions like printf, gotoxy etc while working in graphics mode. Also note '\n' or other escape sequences do not work in graphics mode.

outtextxy function display text or string at a specified point(x,y) on the screen.

Declaration :- void outtextxy(int x, int y, char \*string);  
x, y are coordinates of the point and third argument contains the address of string to be displayed.

## C programming code for outtextxy

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd,&gm,"C:**\\**TC**\\**BGI");

outtextxy(100, 100, "Outtextxy function");

getch();

closegraph();

return 0;

}

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

pieslice(200,200,0,135,100);

getch();

closegraph();

return 0;

}

Declaration :- void putpixel(int x, int y, int color);

For example if we want to draw a GREEN color pixel at (35, 45) then we will write putpixel(35, 35, GREEN); in our c program, putpixel function can be used to draw circles, lines and ellipses using various algorithms.

## C programming code for putpixel

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

putpixel(25, 25, RED);

getch();

closegraph();

return 0;

}

Sector function draws and fills an elliptical pie slice.

Declaration :- void sector( int x, int y, int stangle, int endangle, int xradius, int yradius);

## C programming code for sector

#include <graphics.h>

#include <conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

sector(100, 100, 0, 135, 25, 35);

getch();

closegraph();

return 0;

}

Declaration :- void setbkcolor(int color);

setbkcolor function changes current background color e.g. setbkcolor(YELLLOW) changes the current background color to YELLOW.  
Remember that default drawing color is WHITE and background color is BLACK.

## C programming code for setbkcolor

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

outtext("Press any key to change the background color to GREEN.");

getch();

setbkcolor(GREEN);

getch();

closegraph();

return 0;

}

Declaration :- void setcolor(int color);

In Turbo Graphics each color is assigned a number. Total 16 colors are available. Strictly speaking number of available colors depends on current graphics mode and driver.For Example :- BLACK is assigned 0, RED is assigned 4 etc. setcolor function is used to change the current drawing color.e.g. setcolor(RED) or setcolor(4) changes the current drawing color to RED. Remember that default drawing color is WHITE.

## C programming code for setcolor

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd,&gm,"C:**\\**TC**\\**BGI");

circle(100,100,50); */\* drawn in white color \*/*

setcolor(RED);

circle(200,200,50); */\* drawn in red color \*/*

getch();

closegraph();

return 0;

}

Declaration :- void rectangle(int left, int top, int right, int bottom);

rectangle function is used to draw a rectangle. Coordinates of left top and right bottom corner are required to draw the rectangle. left specifies the X-coordinate of top left corner, top specifies the Y-coordinate of top left corner, right specifies the X-coordinate of right bottom corner, bottom specifies the Y-coordinate of right bottom corner. The code given below draws a rectangle.

## c programming code for rectangle

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

rectangle(100,100,200,200);

getch();

closegraph();

return 0;

}

Declaration:  
void setlinestyle( int linestyle, unsigned upattern, int thickness );

Available line styles:

**enum** line\_styles

{

SOLID\_LINE,

DOTTED\_LINE,

CENTER\_LINE,

DASHED\_LINE,

USERBIT\_LINE

};

## C programming code

#include <graphics.h>

main()

{

int gd = DETECT, gm, c , x = 100, y = 50;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

for ( c = 0 ; c < 5 ; c++ )

{

setlinestyle(c, 0, 2);

line(x, y, x+200, y);

y = y + 25;

}

getch();

closegraph();

return 0;

}

setfillstyle function sets the current fill pattern and fill color.

Declaration :- void setfillstyle( int pattern, int color);

Different fill styles:

**enum** fill\_styles

{

EMPTY\_FILL,

SOLID\_FILL,

LINE\_FILL,

LTSLASH\_FILL,

SLASH\_FILL,

BKSLASH\_FILL,

LTBKSLASH\_FILL,

HATCH\_FILL,

XHATCH\_FILL,

INTERLEAVE\_FILL,

WIDE\_DOT\_FILL,

CLOSE\_DOT\_FILL,

USER\_FILL

};

## C programming source code for setfillstyle

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

setfillstyle(XHATCH\_FILL, RED);

circle(100, 100, 50);

floodfill(100, 100, WHITE);

getch();

closegraph();

return 0;

}

Settextstyle function is used to change the way in which text appears, using it we can modify the size of text, change direction of text and change the font of text.

Declaration :- void settextstyle( int font, int direction, int charsize);  
font argument specifies the font of text, Direction can be HORIZ\_DIR (Left to right) or VERT\_DIR (Bottom to top).

## Different fonts

**enum** font\_names

{

DEFAULT\_FONT,

TRIPLEX\_FONT,

SMALL\_FONT,

SANS\_SERIF\_FONT,

GOTHIC\_FONT,

SCRIPT\_FONT,

SIMPLEX\_FONT,

TRIPLEX\_SCR\_FONT,

COMPLEX\_FONT,

EUROPEAN\_FONT,

BOLD\_FONT

};

### C programming source code for settextstyle

#include <graphics.h>

#include <conio.h>

main()

{

int gd = DETECT, gm, x = 25, y = 25, font = 0;

initgraph(&gd,&gm,"C:**\\**TC**\\**BGI");

for ( font = 0 ; font <= 10 ; font++)

{

settextstyle(font, HORIZ\_DIR, 1);

outtextxy(x, y, "Text with different fonts");

y = y + 25;

}

getch();

closegraph();

return 0;

}

setviewport function sets the current viewport for graphics output.

Declaration :- void setviewport(int left, int top, int right, int bottom, int clip);

setviewport function is used to restrict drawing to a particular portion on the screen. For example setviewport(100 , 100, 200, 200, 1);  
will restrict our drawing activity inside the rectangle(100,100, 200, 200).  
left, top, right, bottom are the coordinates of main diagonal of rectangle in which we wish to restrict our drawing. Also note that the point (left, top) becomes the new origin.

## C programming source code for setviewport

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm, midx, midy;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

midx = getmaxx()/2;

midy = getmaxy()/2;

setviewport(midx - 50, midy - 50, midx + 50, midy + 50, 1);

circle(50, 50, 55);

getch();

closegraph();

return 0;

}

textheight function returns the height of a string in pixels.

Declaration :- int textheight(char \*string);

## C programming source code for textheight

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm, height;

char array[100];

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

height = textheight("C programming");

sprintf(array,"Textheight = %d",height);

outtext(array);

getch();

closegraph();

return 0;

}

textwidth function returns the width of a string in pixels.

Declaration :- int textwidth(char \*string);

## C programming source code for textwidth

#include<graphics.h>

#include<conio.h>

main()

{

int gd = DETECT, gm, width;

char array[100];

initgraph(&gd, &gm, "C:**\\**TC**\\**BG I");

width = textwidth("C programming");

sprintf(array,"Textwidth = %d",width);

outtext(array);

getch();

closegraph();

return 0;

}

This program restricts mouse pointer in a circle i.e you can't move mouse out of a circle. When you try to bring mouse pointer outside the circle, mouse pointer is moved to it's previous location which is inside the circle. Code to restrict mouse in circle is given below :-

## C programming code

#include<graphics.h>

#include<conio.h>

#include<dos.h>

#include<stdlib.h>

#include<math.h>

union REGS i, o;

int initmouse()

{

i.x.ax = 0;

int86(0X33, &i, &o);

return ( o.x.ax );

}

void showmouseptr()

{

i.x.ax = 1;

int86(0X33, &i, &o);

}

void hidemopuseptr()

{

i.x.ax = 2;

int86(0X33,&i,&o);

}

void getmousepos(int \*x, int \*y)

{

i.x.ax = 3;

int86(0X33, &i, &o);

\*x = o.x.cx;

\*y = o.x.dx;

}

void movemouseptr(int x, int y)

{

i.x.ax = 4;

i.x.cx = x;

i.x.dx = y;

int86(0X33, &i, &o);

}

main()

{

int gd = DETECT, gm, midx, midy, radius, x, y, tempx, tempy;

radius = 100;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

if(!initmouse())

{

closegraph();

exit(1);

}

midx = getmaxx()/2;

midy = getmaxy()/2;

showmouseptr();

movemouseptr(midx, midy);

circle(midx, midy, radius);

x = tempx = midx;

y = tempy = midy;

while(!kbhit())

{

getmousepos(&x, &y);

if((pow(x-midx,2)+pow(y-midy,2)-pow(radius,2))>0)

{

movemouseptr(tempx, tempy);

x = tempx;

y = tempy;

}

tempx = x;

tempy = y;

}

closegraph();

return 0;

}

Traffic light Simulation: Traffic light program in c presents what happens in our daily life at traffic light signals. Firstly user will press a key to start the traffic light simulation.

## C programming code

#include<graphics.h>

#include<conio.h>

#include<dos.h>

#include<stdlib.h>

main()

{

int gd = DETECT, gm, midx, midy;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

midx = getmaxx()/2;

midy = getmaxy()/2;

setcolor(RED);

settextstyle(SCRIPT\_FONT, HORIZ\_DIR, 3);

settextjustify(CENTER\_TEXT, CENTER\_TEXT);

outtextxy(midx, midy-10, "Traffic Light Simulation");

outtextxy(midx, midy+10, "Press any key to start");

getch();

cleardevice();

setcolor(WHITE);

settextstyle(DEFAULT\_FONT, HORIZ\_DIR, 1);

rectangle(midx-30,midy-80,midx+30,midy+80);

circle(midx, midy-50, 22);

setfillstyle(SOLID\_FILL,RED);

floodfill(midx, midy-50,WHITE);

setcolor(BLUE);

outtextxy(midx,midy-50,"STOP");

delay(2000);

graphdefaults();

cleardevice();

setcolor(WHITE);

rectangle(midx-30,midy-80,midx+30,midy+80);

circle(midx, midy, 20);

setfillstyle(SOLID\_FILL,YELLOW);

floodfill(midx, midy,WHITE);

setcolor(BLUE);

outtextxy(midx-18,midy-3,"READY");

delay(2000);

cleardevice();

setcolor(WHITE);

rectangle(midx-30,midy-80,midx+30,midy+80);

circle(midx, midy+50, 22);

setfillstyle(SOLID\_FILL,GREEN);

floodfill(midx, midy+50,WHITE);

setcolor(BLUE);

outtextxy(midx-7,midy+48,"GO");

setcolor(RED);

settextstyle(SCRIPT\_FONT, HORIZ\_DIR, 4);

outtextxy(midx-150, midy+100, "Press any key to exit...");

getch();

closegraph();

return 0;

}

Press me game in C:- In this game when you try to bring mouse near a button it moves away from the mouse, so you keep on trying pressing the button. In this game we have the coordinates of current position of mouse pointer at every moment of time, whenever we find mouse pointer very close to button we move the button suitably.

## C programming code

#include <stdio.h>

#include <conio.h>

#include <dos.h>

#include <graphics.h>

#include <stdlib.h>

union REGS i, o;

int left = 265, top = 250;

void initialize\_graphics\_mode()

{

int gd = DETECT, gm, error;

initgraph(&gd,&gm,"C:**\\**TC**\\**BGI");

error = graphresult();

if (error != grOk)

{

perror("Error ");

printf("Press any key to exit...**\n**");

getch();

exit(EXIT\_FAILURE);

}

}

void showmouseptr()

{

i.x.ax = 1;

int86(0x33,&i,&o);

}

void hidemouseptr()

{

i.x.ax = 2;

int86(0x33,&i,&o);

}

void getmousepos(int \*x,int \*y)

{

i.x.ax = 3;

int86(0x33,&i,&o);

\*x = o.x.cx;

\*y = o.x.dx;

}

void draw\_bar()

{

hidemouseptr();

setfillstyle(SOLID\_FILL,CYAN);

bar(190,180,450,350);

showmouseptr();

}

void draw\_button(int x, int y)

{

hidemouseptr();

setfillstyle(SOLID\_FILL,MAGENTA);

bar(x,y,x+100,y+30);

moveto(x+5,y);

setcolor(YELLOW);

outtext("Press me");

showmouseptr();

}

void draw()

{

settextstyle(SANS\_SERIF\_FONT,HORIZ\_DIR,2);

outtextxy(155,451,"<a href="http:*//www.programmingsimplified.com"">www.programmingsimplified.com"</a>);*

setcolor(BLUE);

rectangle(0,0,639,450);

setcolor(RED);

outtextxy(160,25,"Try to press the **\"**Press me**\"** button");

outtextxy(210,50,"Press escape key to exit");

setfillstyle(XHATCH\_FILL,GREEN);

setcolor(BLUE);

bar(1,1,75,449);

bar(565,1,638,449);

showmouseptr();

draw\_bar();

draw\_button(left,top);

}

void initialize()

{

initialize\_graphics\_mode();

if( !initmouse() )

{

closegraph();

printf("Unable to initialize the mouse");

printf("Press any key to exit...**\n**");

getch();

exit(EXIT\_SUCCESS);

}

draw();

}

int initmouse()

{

i.x.ax = 0;

int86(0x33,&i,&o);

return ( o.x.ax );

}

void get\_input()

{

int x, y;

while(1)

{

getmousepos(&x,&y);

*/\* mouse pointer in left of button \*/*

if( x >= (left-3) && y >= (top-3) && y <= (top+30+3) && x < left )

{

draw\_bar();

left = left + 4;

if (left > 350)

left = 190;

draw\_button(left,top);

}

*/\* mouse pointer in right of button \*/*

else if (x<=(left+100+3)&&y>=(top-3)&&y<=(top+30+3)&&x>(left+100))

{

draw\_bar();

left = left - 4;

if (left < 190)

left = 350;

draw\_button(left,top);

}

*/\* mouse pointer above button \*/*

else if(x>(left-3) && y>=(top-3) && y<(top) && x<= (left+100+3))

{

draw\_bar();

top = top + 4;

if (top > 320)

top = 180;

draw\_button(left,top);

}

*/\* mouse pointer below button \*/*

else if (x>(left-3)&&y>(top+30)&&y<=(top+30+3)&&x<=(left+100+3))

{

draw\_bar();

top = top - 4;

if (top < 180)

top = 320;

draw\_button(left,top);

}

if (kbhit())

{

if (getkey() == 1)

exit(EXIT\_SUCCESS);

}

}

}

int getkey()

{

i.h.ah = 0;

int86(22,&i,&o);

return( o.h.ah );

}

main()

{

initialize();

get\_input();

return 0;

}

Paint program in c:- This program can draw different shapes using mouse such as line, circle, pixel and many other shapes. You can also change the color, clear the screen. Code of paint program in c is given below:-

/\* To understand the code see output below the code, it will help you in understanding the code. \*/

**C programming code**

#include<graphics.h>

#include<dos.h>

#include<math.h>

#include<stdlib.h>

#include<stdio.h>

#include<conio.h>

union REGS i, o;

int leftcolor[15];

int get\_key()

{

union REGS i,o;

i.h.ah = 0;

int86(22,&i,&o);

return ( o.h.ah );

}

void draw\_color\_panel()

{

int left, top, c, color;

left = 100;

top = 436;

color = getcolor();

setcolor(GREEN);

rectangle(4,431,635,457);

setcolor(RED);

settextstyle(TRIPLEX\_FONT,0,2);

outtextxy(10,431,"Colors : ");

for( c = 1 ; c <= 15 ; c++ )

{

setfillstyle(SOLID\_FILL, c);

bar(left, top, left+16, top+16);

leftcolor[c-1] = left;

left += 26;

}

setcolor(color);

}

void draw\_shape\_panel()

{

int left, top, c, color;

left = 529;

top = 45;

color = getcolor();

setcolor(GREEN);

rectangle(525,40,633,255);

for( c = 1 ; c <= 7 ; c++ )

{

rectangle(left, top, left+100, top+25);

top += 30;

}

setcolor(RED);

outtextxy(530,45,"Bar");

outtextxy(530,75,"Line");

outtextxy(530,105,"Pixel");

outtextxy(530,135,"Ellipse");

outtextxy(530,165,"Freehand");

outtextxy(530,195,"Rectangle");

outtextxy(530,225,"Clear");

setcolor(color);

}

void change\_color(int x, int y)

{

int c;

for( c = 0 ; c <= 13 ; c++ )

{

if( x > leftcolor[c] && x < leftcolor[c+1] && y > 437 && y < 453 )

setcolor(c+1);

if( x > leftcolor[14] && x < 505 && y > 437 && y < 453 )

setcolor(WHITE);

}

}

char change\_shape(int x, int y)

{

if ( x > 529 && x < 625 && y > 45 && y < 70 )

return 'b';

else if ( x > 529 && x < 625 && y > 75 && y < 100 )

return 'l';

else if ( x > 529 && x < 625 && y > 105 && y < 130 )

return 'p';

else if ( x > 529 && x < 625 && y > 135 && y < 160 )

return 'e';

else if ( x > 529 && x < 625 && y > 165 && y < 190 )

return 'f';

else if ( x > 529 && x < 625 && y > 195 && y < 220 )

return 'r';

else if ( x > 529 && x < 625 && y > 225 && y < 250 )

return 'c';

}

void showmouseptr()

{

i.x.ax = 1;

int86(0x33,&i,&o);

}

void hidemouseptr()

{

i.x.ax = 2;

int86(0x33,&i,&o);

}

void restrictmouseptr( int x1, int y1, int x2, int y2)

{

i.x.ax = 7;

i.x.cx = x1;

i.x.dx = x2;

int86(0x33,&i,&o);

i.x.ax = 8;

i.x.cx = y1;

i.x.dx = y2;

int86(0x33,&i,&o);

}

void getmousepos(int \*button,int \*x,int \*y)

{

i.x.ax = 3;

int86(0x33,&i,&o);

\*button = o.x.bx;

\*x = o.x.cx;

\*y = o.x.dx;

}

main()

{

int gd = DETECT,gm;

int maxx,maxy,x,y,button,prevx,prevy,temp1,temp2,key,color;

char ch = 'f' ; *// default free-hand drawing*

initgraph(&gd,&gm,"C:**\\**TC**\\**BGI");

maxx = getmaxx();

maxy = getmaxy();

setcolor(BLUE);

rectangle(0,0,maxx,maxy);

setcolor(WHITE);

settextstyle(SANS\_SERIF\_FONT,HORIZ\_DIR,2);

outtextxy(maxx/2-180,maxy-28,"<a href="http:*//www.programmingsimplified.com"">www.programmingsimplified.com"</a>);*

draw\_color\_panel();

draw\_shape\_panel();

setviewport(1,1,maxx-1,maxy-1,1);

restrictmouseptr(1,1,maxx-1,maxy-1);

showmouseptr();

rectangle(2,2,518,427);

setviewport(1,1,519,428,1);

while(1)

{

if(kbhit())

{

key = get\_key();

if( key == 1 )

{

closegraph();

exit(0);

}

}

getmousepos(&button,&x,&y);

if( button == 1 )

{

if( x > 4 && x < 635 && y > 431 && y < 457 )

change\_color( x, y );

else if ( x > 529 && x < 625 && y > 40 && y < 250 )

ch = change\_shape( x, y );

temp1 = x ;

temp2 = y ;

if ( ch == 'f' )

{

hidemouseptr();

while( button == 1 )

{

line(temp1,temp2,x,y);

temp1 = x;

temp2 = y;

getmousepos(&button,&x,&y);

}

showmouseptr();

}

while( button == 1)

getmousepos(&button,&x,&y);

*/\* to avoid interference of mouse while drawing \*/*

hidemouseptr();

if( ch == 'p')

putpixel(x,y,getcolor());

else if ( ch == 'b' )

{

setfillstyle(SOLID\_FILL,getcolor());

bar(temp1,temp2,x,y);

}

else if ( ch == 'l')

line(temp1,temp2,x,y);

else if ( ch == 'e')

ellipse(temp1,temp2,0,360,abs(x-temp1),abs(y-temp2));

else if ( ch == 'r' )

rectangle(temp1,temp2,x,y);

else if ( ch == 'c' )

{

ch = 'f'; *// setting to freehand drawing*

clearviewport();

color = getcolor();

setcolor(WHITE);

rectangle(2,2,518,427);

setcolor(color);

}

showmouseptr();

}

}

}

This c graphics program performs countdown for 30 seconds.

## C program countdown code

#include<graphics.h>

#include<dos.h>

#include<conio.h>

main()

{

int gd = DETECT, gm, i;

char a[5];

initgraph( &gd, &gm, "C:**\\**TC**\\**BGI");

settextjustify( CENTER\_TEXT, CENTER\_TEXT );

settextstyle(DEFAULT\_FONT,HORIZ\_DIR,3);

setcolor(RED);

for(i = 30 ; i >=0 ; i--)

{

sprintf(a,"%d",i);

outtextxy(getmaxx()/2, getmaxy()/2, a);

delay(1000);

if ( i == 0 )

**break**;

cleardevice();

}

getch();

closegraph();

return 0;

}

This program opens a website entered by the user. User can open any website .It will launch Mozilla Firefox web browser to open a website so it should be installed on your computer, if you are using an another web browser then you can change the path in the program.

## C programming code

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<graphics.h>

#include<dos.h>

#include<string.h>

void initialize\_graphics\_mode();

int get\_key();

void draw();

union REGS i, o;

main()

{

int key, i = 0, xpos, ypos, button;

char arr[200], temp[5], \*ptr;

char a[] = "C:**\\**Progra~1**\\**Mozill~1**\\**firefox ";

strcpy(arr,a);

i = strlen(a);

initialize\_graphics\_mode();

draw();

while(1)

{

if(kbhit())

key = get\_key();

if((key>=97&&key<=122)||(key>=65&&key<=90)||key==46||key==47||key==63)

{

arr[i] = key;

sprintf(temp,"%c",arr[i]);

outtext(temp);

if(getx()>470)

{

clearviewport();

moveto(5,2);

}

i++;

}

else if ( key == 13 )

{

arr[i] = '**\0**';

system(arr);

**break**;

}

else if ( key == 27 )

{

closegraph();

exit(EXIT\_SUCCESS);

}

if(button==1&&xpos>=150&&xpos<=480&&ypos>=300&&ypos<=330)

{

system("C:**\\**Progra~1**\\**Mozill~1**\\**firefox programmingsimplified.com");

**break**;

}

key = -1;

}

closegraph();

return 0;

}

void initialize\_graphics\_mode()

{

int gd = DETECT, gm, errorcode;

initgraph(&gd,&gm,"C:**\\**TC**\\**BGI");

errorcode = graphresult();

if( errorcode != grOk )

{

printf("Graphics error : %s**\n**",grapherrormsg(errorcode));

printf("Press any key to exit...**\n**");

getch();

exit(EXIT\_FAILURE);

}

}

int get\_key()

{

i.h.ah = 0;

int86(22,&i,&o);

return( o.h.al );

}

void draw()

{

settextstyle(SANS\_SERIF\_FONT,HORIZ\_DIR,2);

outtextxy(275,11,"Web Browser");

outtextxy(155,451,"<a href="http:*//www.programmingsimplified.com"">www.programmingsimplified.com"</a>);*

outtextxy(5,105,"Enter URL : ");

rectangle(120,100,600,130);

setviewport(121,101,599,129,1);

moveto(5,1);

}

This program draws circles in circles in two different colors.

## C programming code

#include<graphics.h>

#include<conio.h>

#include<dos.h>

main()

{

int gd = DETECT, gm, x, y, color, angle = 0;

struct arccoordstype a, b;

initgraph(&gd, &gm, "C:**\\**TC**\\**BGI");

delay(2000);

while(angle<=360)

{

setcolor(BLACK);

arc(getmaxx()/2,getmaxy()/2,angle,angle+2,100);

setcolor(RED);

getarccoords(&a);

circle(a.xstart,a.ystart,25);

setcolor(BLACK);

arc(getmaxx()/2,getmaxy()/2,angle,angle+2,150);

getarccoords(&a);

setcolor(GREEN);

circle(a.xstart,a.ystart,25);

angle = angle+5;

delay(50);

}

getch();

closegraph();

return 0;

}

This program generates captcha, a captcha is a random code generated using some algorithm. We will use random function in our code. These are used in typing tutors and in website to check whether a human is operating on a website.

**C programming code**

#include<stdlib.h>

#include<dos.h>

#include<graphics.h>

main()

{

int i = 0, key, num, midx, gd = DETECT, gm;

char a[10];

initgraph(&gd,&gm,"C:**\\**TC**\\**BGI");

midx = getmaxx()/2;

settextstyle(SCRIPT\_FONT,HORIZ\_DIR,5);

settextjustify(CENTER\_TEXT,CENTER\_TEXT);

setcolor(GREEN);

outtextxy(midx,20,"CAPTCHA");

settextstyle(SCRIPT\_FONT,HORIZ\_DIR,2);

outtextxy(midx,125,"Press any key to change the generated random code **\"**captcha**\"**");

outtextxy(midx,150,"Press escape key to exit...");

setcolor(WHITE);

setviewport(100,200,600,400,1);

setcolor(RED);

randomize();

while(1)

{

while(i<6)

{

num = random(3);

if ( num == 0 )

a[i] = 65 + random(26); */\* 65 is the ASCII value of A \*/*

else if ( num == 1)

a[i] = 97 + random(26); */\* 97 is the ASCII value of a \*/*

else

a[i] = 48 + random(10); */\* 48 is the ASCII value of 0 \*/*

i++;

}

a[i] = '**\0**';

outtextxy(210,100,a);

key = getch();

if( key == 27 ) */\* escape key\*/*

exit(0);

clearviewport();

i = 0;

}

}

/\* Program to check if mouse driver is loaded or not \*/

## C programming code

#include<dos.h>

#include<conio.h>

int initmouse();

union REGS i, o;

main()

{

int status;

status = initmouse();

if ( status == 0 )

printf("Mouse support not available.**\n**");

else

printf("Mouse support available.**\n**");

getch();

return 0;

}

int initmouse()

{

i.x.ax = 0;

int86(0X33,&i,&o);

return ( o.x.ax );

}

/\* Program to display mouse-pointer in text-mode \*/

#include<dos.h>

#include<conio.h>

int initmouse();

void showmouseptr();

union REGS i, o;

main()

{

int status;

status = initmouse();

if ( status == 0 )

printf("Mouse support not available.**\n**");

else

showmouseptr();

getch();

return 0;

}

int initmouse()

{

i.x.ax = 0;

int86(0X33,&i,&o);

return ( o.x.ax );

}

void showmouseptr()

{

i.x.ax = 1;

int86(0X33,&i,&o);

}

This program displays mouse pointer in graphics mode. First graphics mode is initialized and then mouse using initmouse.

## C programming code

#include<graphics.h>

#include<conio.h>

#include<dos.h>

int initmouse();

void showmouseptr();

union REGS i, o;

main()

{

int status, gd = DETECT, gm;

initgraph(&gd,&gm,"C:**\\**TC**\\**BGI");

status = initmouse();

if ( status == 0 )

printf("Mouse support not available.**\n**");

else

showmouseptr();

getch();

return 0;

}

int initmouse()

{

i.x.ax = 0;

int86(0X33,&i,&o);

return ( o.x.ax );

}

void showmouseptr()

{

i.x.ax = 1;

int86(0X33,&i,&o);

}

This program hides mouse pointer. We require to hide mouse pointer when we want to draw something on screen as it may interfere with drawing, after that we again make it visible.  
/\* Program to show and hide mouse-pointer alternatively \*/

## C programming code

#include<graphics.h>

#include<conio.h>

#include<dos.h>

int initmouse();

void showmouseptr();

void hidemouseptr();

union REGS i, o;

main()

{

int status, count = 1, gd = DETECT, gm;

initgraph(&gd,&gm,"C:**\\**TC**\\**BGI");

status = initmouse();

if ( status == 0 )

printf("Mouse support not available.**\n**");

else

{

showmouseptr();

while(count<=10)

{

getch();

count++;

if(count%2==0)

hidemouseptr();

else

showmouseptr();

}

}

getch();

return 0;

}

int initmouse()

{

i.x.ax = 0;

int86(0X33,&i,&o);

return ( o.x.ax );

}

void showmouseptr()

{

i.x.ax = 1;

int86(0X33,&i,&o);

}

void hidemouseptr()

{

i.x.ax = 2; *// to hide mouse*

int86(0X33,&i,&o);

}