1.

Arc function in c

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.

```
#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;
}
```

In the above program (100,100) are coordinates of center of arc, 0 is the starting angle, 135 is the end angle and 50 specifies the radius of the arc.

2.

Bar function in c

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.

```
#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;
}
```

3.

bar3d function in c

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.

```
#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;
}
```

4.

Circle function in c

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.

```
#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.

5.

Cleardevice function in c

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.

```
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;
}
```

Note : Don't use clrscr in graphics mode.

6.

closegraph function in c

closegraph function closes the graphics mode, deallocates all memory allocated by graphics system and restores the screen to the mode it was in before you called initgraph.

Declaration :- void closegraph();

```
#include<graphics.h>
#include<conio.h>

main()
{
    int gd = DETECT, gm;

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

    outtext("Press any key to close the graphics mode...");

    getch();
    closegraph();
    return 0;
}
```

drawpoly function in c

Drawpoly function is used to draw polygons i.e. triangle, 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.

```
#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;
}
```

7.

ellipse function in c

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.

```
#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;
}
```

8.

fillellipse function in c

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.

```
#include <graphics.h>
#include <conio.h>

int main()
{
   int gd = DETECT, gm;

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

   fillellipse(100, 100, 50, 25);

   getch();
   closegraph();
   return 0;
}
```

9.

fillpoly function in c

Fillpoly function draws and fills a polygon. It require same arguments as drawpoly.

Declaration :- void drawpoly(int num, int *polypoints);
For details of arguments see drawpoly.
fillpoly fills using current fill pattern and color which can be changed using setfillstyle.

```
#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;
}
```

10.

Floodfill function

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.

```
#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.

11.

Getarcoords function in c

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.

```
#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 outtextxy.

12.

getbkcolor function in c

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.

```
#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;
}
```

13.

getcolor function

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.

```
#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;
}
```

14.

colors in c graphics.

Following colors are available for use in c graphics programming.

Colors table

Color	Value
BLACK	0
BLUE	1
GREEN	2
CYAN	3
RED	4

	MAGENTA	5
	BROWN	6
	LIGHTGRAY	7
	DARKGRAY	8
	LIGHTBLUE	9
	LIGHTGREEN	10
	LIGHTCYAN	11
	LIGHTRED	12
	LIGHTMAGENTA	13
	YELLOW	14
	WHITE	15

Total number of colors available depend on current graphics driver and mode. Use colors name in capital letters, for example use setcolor(RED) not setcolor(red) the latter will give you an error. You may use number instead of color for example setbkcolor(GREEN) or setbkcolor(2) are same, but you are advised to use color name as it will improve readability of program.

15.

getimage function in c

getimage function saves a bit image of specified region into memory, region can be any rectangle.

Declaration:- void getimage(int left, int top, int right, int bottom, void *bitmap);

getimage copies an image from screen to memory. Left, top, right, and bottom define the area of the screen from which the rectangle is to be copied, bitmap points to the area in memory where the bit image is stored.

Smiling face animation program using getimage.

C smiling face animation

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.

```
#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;
}
```

16.

getmaxcolor function

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();

```
#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;
}
```

17.

getmaxx function in c

getmaxx function returns the maximum X coordinate for current graphics mode and driver.

Declaration :- int getmaxx();

```
#include<graphics.h>
#include<conio.h>

main()
{
   int gd = DETECT, gm, max_x;
   char array[100];

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

   max_x = getmaxx();

   sprintf(array, "Maximum X coordinate for current graphics mode and driver =
%d.",max_x);
   outtext(array);

   getch();
   closegraph();
   return 0;
}
```

getmaxy function in c

getmaxy function returns the maximum Y coordinate for current graphics mode and driver.

Declaration :- int getmaxy();

```
#include<graphics.h>
#include<conio.h>

main()
{
   int gd = DETECT, gm, max_y;
   char array[100];

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

   max_y = getmaxy();

   sprintf(array, "Maximum Y coordinate for current graphics mode and driver is =
%d.",max_y);
   outtext(array);

   getch();
   closegraph();
   return 0;
}
```

18.

getpixel function in c

getpixel function returns the color of pixel present at location(x, y).

Declaration :- int getpixel(int x, int y);

```
#include<graphics.h>
#include<conio.h>

main()
{
  int gd = DETECT, gm, color;
  char array[50];

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

  color = getpixel(0, 0);

  sprintf(array,"color of pixel at (0,0) = %d",color);
  outtext(array);

  getch();
  closegraph();
  return 0;
}
```

As we haven't drawn anything on screen and by default screen is BLACK, therefore color of pixel at (0,0) is BLACK. So output of program will be color of pixel at (0,0) is 0, as 0 indicates BLACK color.

19.

getx function in c

getx function returns the X coordinate of current position.

Declaration :- int 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;
}
```

20.

gety function in c

gety function returns the y coordinate of current position.

Declaration :- int 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;
}
```

21.

graphdefaults function in c

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.

```
#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. See colors in c graphics.

22.

grapherrormsg function in c

grapherrormsg function returns an error message string.

Declaration :- char *grapherrormsg(int errorcode);

```
#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.
After running this program we get the output :-
Graphics error: Graphics hardware not detected
Press any key to exit.

23.

imagesize function in c

imagesize function returns the number of bytes required to store a bitimage. This function is used when we are using getimage.

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

```
#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;
}
```

24.

Line function in c

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);

```
#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;
}
```

Above program draws a line from (100, 100) to (200, 200).

25.

lineto function in c

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

```
#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;
}
```

26.

linerel function in c

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 and gety to find the current position.

```
#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;
}
```

27.

moveto function in c

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

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

```
#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;
}
```

28.

moverel function in c

moverel function moves the current position to a relative distance.

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

```
#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;
}
```

29.

outtext function

outtext function displays text at current position.

Declaration :- void outtext(char *string);

```
#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. You have to ensure that the text doesn't go beyond the screen while using outtext.

30.

outtextxy function in c

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.

```
#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;
}
```

31.

pieslice function in c

```
#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;
}
```

32.

putimage function in c

putimage function outputs a bit image onto the screen.

Declaration:- void putimage(int left, int top, void *ptr, int op);

putimage puts the bit image previously saved with getimage back onto the screen, with the upper left corner of the image placed at (left, top). ptr points to the area in memory where the source image is stored. The op argument specifies a operator that controls how the color for each destination pixel on screen is computed, based on pixel already on screen and the corresponding source pixel in memory.

Smiling face animation program using putimage.

33.

putpixel function in c

putpixel function plots a pixel at location (x, y) of specified color.

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.

```
#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;
}
```

34.

rectangle function in c

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.

```
#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;
}
```

35.

sector function in c

Sector function draws and fills an elliptical pie slice.

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

```
#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;
}
```

36.

Setbkcolor function in c

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.

```
#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;
}
```

37.

setcolor function in c

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.

```
#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;
}
```

38.

setfillstyle function in c

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
};
#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;
}
```

39.

setlinestyle in c

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
};
#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;
}
```

40.

Settextstyle function in c

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).

```
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
};
#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;
}
```

41.

setviewport function in c

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.

```
#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;
}
```

42.

textheight function in c

textheight function returns the height of a string in pixels.

Declaration :- int textheight(char *string);

```
#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;
}
```

43.

textwidth function in c

textwidth function returns the width of a string in pixels.

Declaration :- int textwidth(char *string);

```
#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;
}
```