Introduction to Graphics

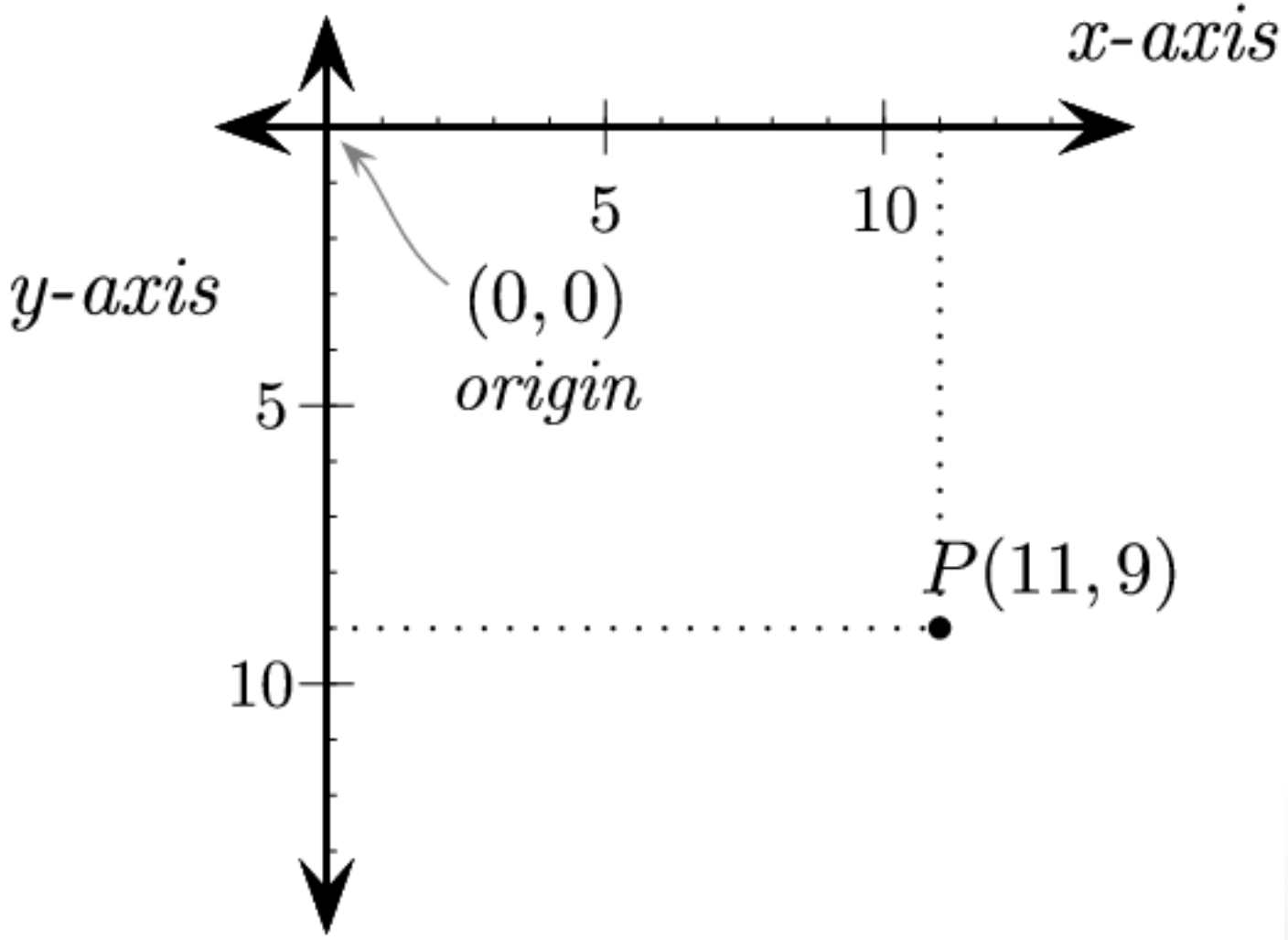
Unit 11

11.1 Concept Of Graphics

- C graphics using **graphics.h** functions can be used to draw different shapes, display text in different fonts , change color and many more.
- Using functions of **graphics.h** in compiler we can make graphics programs, animations, projects, and games.
- We can draw circles, lines, rectangles, bars and many other geometrical figures.
- We can change their colors using the available functions and fill them.

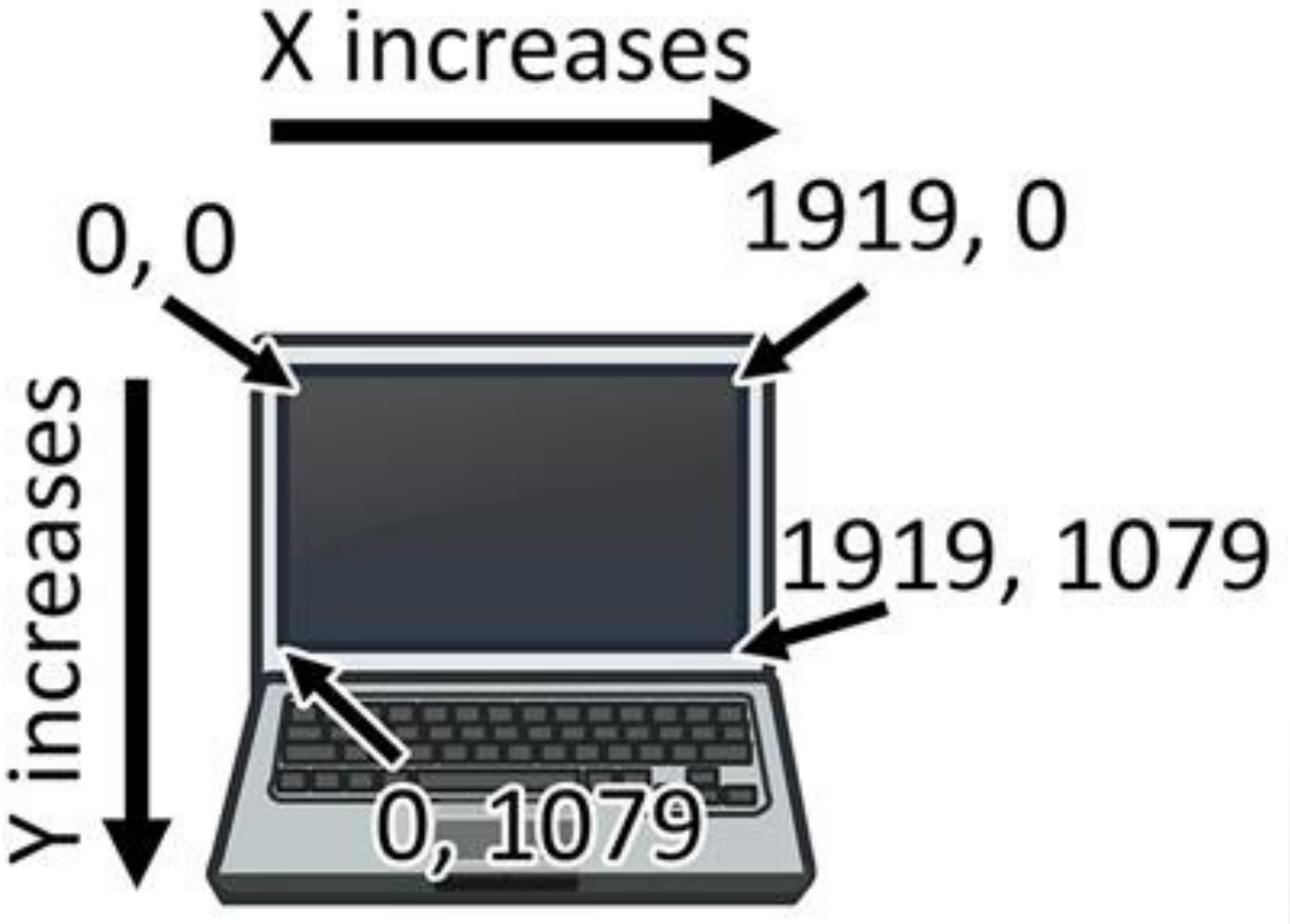
11.1 Concept Of Graphics

Co-ordinate System



11.1 Concept Of Graphics

Co-ordinate System



11.2 Graphics Initialization and Modes

This contains has lots of fundamental graphics program like drawing of various geometrical shapes (**rectangle**, **circle, eclipse** etc), use of mathematical function in drawing curves, coloring an object with different **colors** and patterns and simple animation programs like jumping ball and moving cars.

This chapter will cover an overview of computer graphics and it's fundamentals.

11.2 Graphics Initialization and Modes…..

The first step in any graphics program is to initialize the graphics drivers on the computer using **initgraph** method of **graphics.h** library.

void initgraph (int *graphicsDriver, int *graphicsMode, char *driverDirectoryPath);

11.2 Graphics Initialization and Modes…..

- **initgraph** initializes the graphics system by loading a graphics driver from disk (or validating a registered driver), and putting the system into graphics mode.

11.2 Graphics Initialization and Modes…..

- To start the graphics system, first call the **initgraph** function.
- initgraph loads the graphics driver and puts the system into graphics mode. we can tell **initgraph** to use a particular graphics driver and mode, or to **autodetect** the attached video adapter at run time and pick the corresponding driver.

11.2 Graphics Initialization and Modes…..

- If we tell **initgraph** to **autodetect**, it calls **detectgraph** to select a graphics driver and mode.
- **initgraph** also resets all graphics settings to their defaults (current position, palette, color, viewport, and so on) and resets graph result to 0.

11.2 Graphics Initialization and Modes…..

graphicsDriver :

It is a pointer to an integer specifying the graphics driver to be used.

It tells the compiler that what graphics driver to use or to automatically detect the drive. In all our programs we will use **DETECT** macro of **graphics.h** library that instruct compiler for auto detection of graphics driver.

11.2 Graphics Initialization and Modes…..

graphicsMode :

It is a pointer to an integer that specifies the graphics mode to be used.

If *graphdriver is set to DETECT, then initgraph sets *graphmode to the highest resolution available for the detected driver.

11.2 Graphics Initialization and Modes…..

driverDirectoryPath :

It specifies the directory path where graphics driver files (BGI files) are located.

If directory path is not provided, then it will search for driver files in current working directory directory. In all our sample graphics programs, we have to change path of BGI directory accordingly where our turbo C compiler is installed.

11.2 Graphics Initialization and Modes…..

Example 1.

A Simple Program

```
#include <graphics.h>
#include<conio.h>
void main()
{
        int gd = DETECT, gm;
        initgraph(&gd, &gm, "c:\\tc\\bgi");
        //Your Code goes Here
        getch();
         closegraph();
}
```

11.2 Graphics Initialization and Modes…..

Output of this simple program

1. This program initializes graphics mode and then closes it after a key is pressed.
2. To begin with we have declared two variables of int type **gd** and **gm** for graphics driver and graphics mode respectively.
3. DETECT is a macro defined in "**graphics.h**" header file.
4. Then we have passed three arguments to initgraph function 1st is the address of gd, 2nd is the address of gm and 3rd is the path where your BGI files are present
5. **getch** helps us to wait until a key is pressed, **closegraph** function closes the graphics mode and finally return statement returns a value 0 to main indicating successful execution of your program

11.2 Graphics Initialization and Modes…..

- **outtextxy** ()
  - Outtextxy display a string at the specified location (Graphics mode)

Declaration :

void outtextxy(x,y,"text string");

11.2 Graphics Initialization and Modes…..

Program---Printing Text in Graphics Using Outtextxy Function

Example 2.

```
#include<graphics.h>
 #include<stdio.h>
void main()
{
        int gdriver = DETECT, gmode;
        int x = 200, y = 200;
        initgraph(&gdriver, &gmode, "c:\\tc\\bgi");
        outtextxy(x, y, "Hello NCC");
        getch();
        closegraph();
}
```

11.2 Graphics Initialization and Modes…..

Explanation

outtextxy(x,y,"Hello World");

This Function is Similar to Printf Statement.

Printf Prints Text on Screen in “Text Mode” while outtextxy() function Prints Text onto Screen in “Graphics Mode“.

This Function Accepts 3 Parameters.

Syntax: **outtextxy(x,y**,"Hello World");

11.2 Graphics Initialization and Modes…..

Output

// Colors in C Graphics Programming

There are 16 colors declared in C Graphics.

We use colors to set the current drawing color, change the color of background, change the color of text, to color a closed shape etc.

To specify a color, we can either use color constants like **setcolor(RED)**, or their corresponding integer codes like **setcolor(4)** . Below is the color code in increasing order.

// Colors in C Graphics Programming...

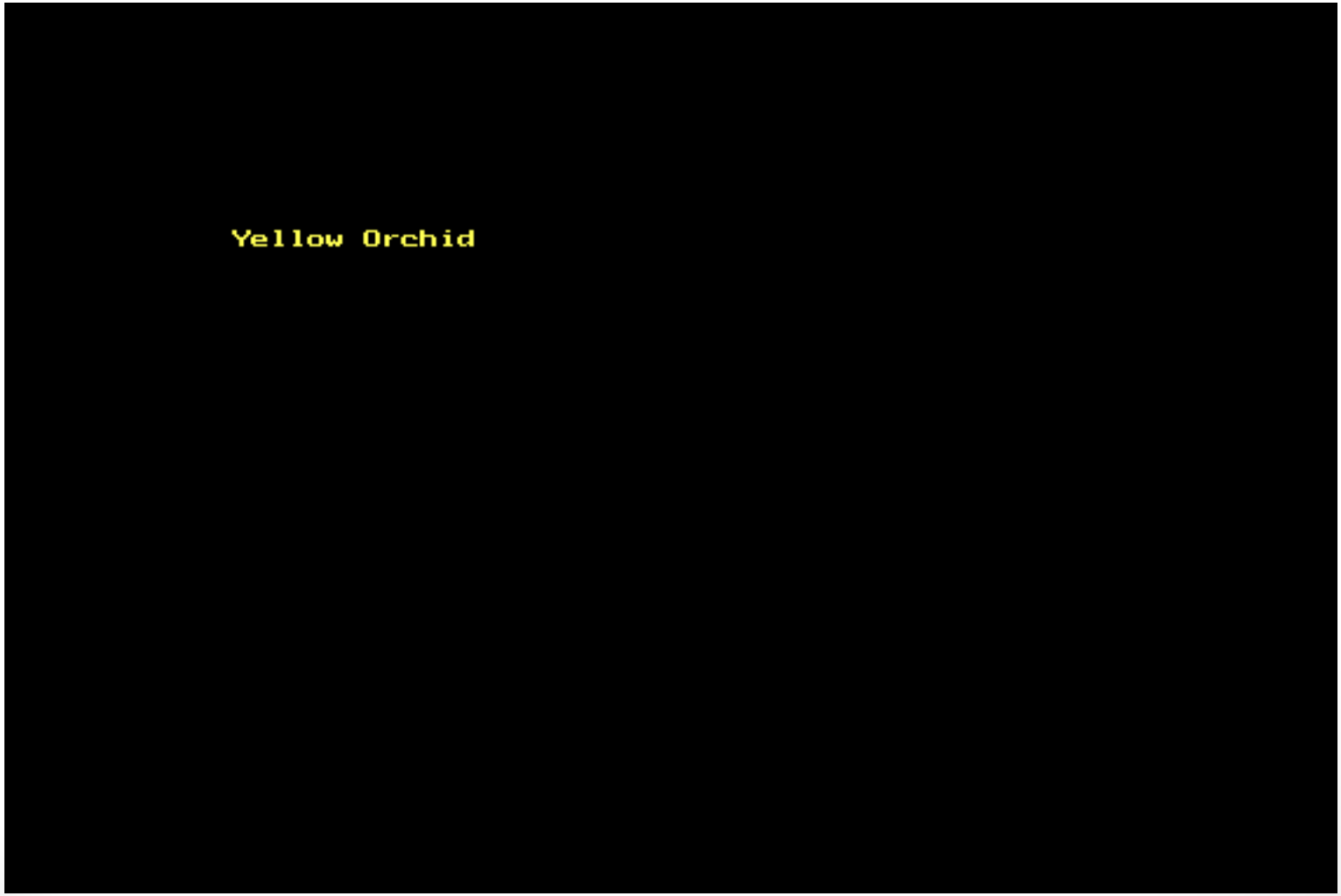
COLOR MACRO	INTEGER VALUE
BLACK	0
BLUE	1
GREEN	2
CYAN	3
RED	4
MAGENTA	5
BROWN	6
LIGHTGRAY	7
DARKGRAY	8
LIGHTBLUE	9
LIGHTGREEN	10
LIGHTCYAN	11
LIGHTRED	12
LIGHTMAGENTA	13
YELLOW	14
WHITE	15

// Colors in C Graphics Programming…

Example 3.

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

void main()
{
        int gd = DETECT, gm;
        initgraph(&gd, &gm, "C:\\TC\\BGI");

        setcolor(YELLOW);
        outtextxy(100,100,"Yellow Orchid");

        getch();
        closegraph();

}
```

// Colors in C Graphics Programming…



11.3 Graphics Function…

- **Line()**

  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.

Declaration:

void line(int x1, int y1, int x2, int y2);

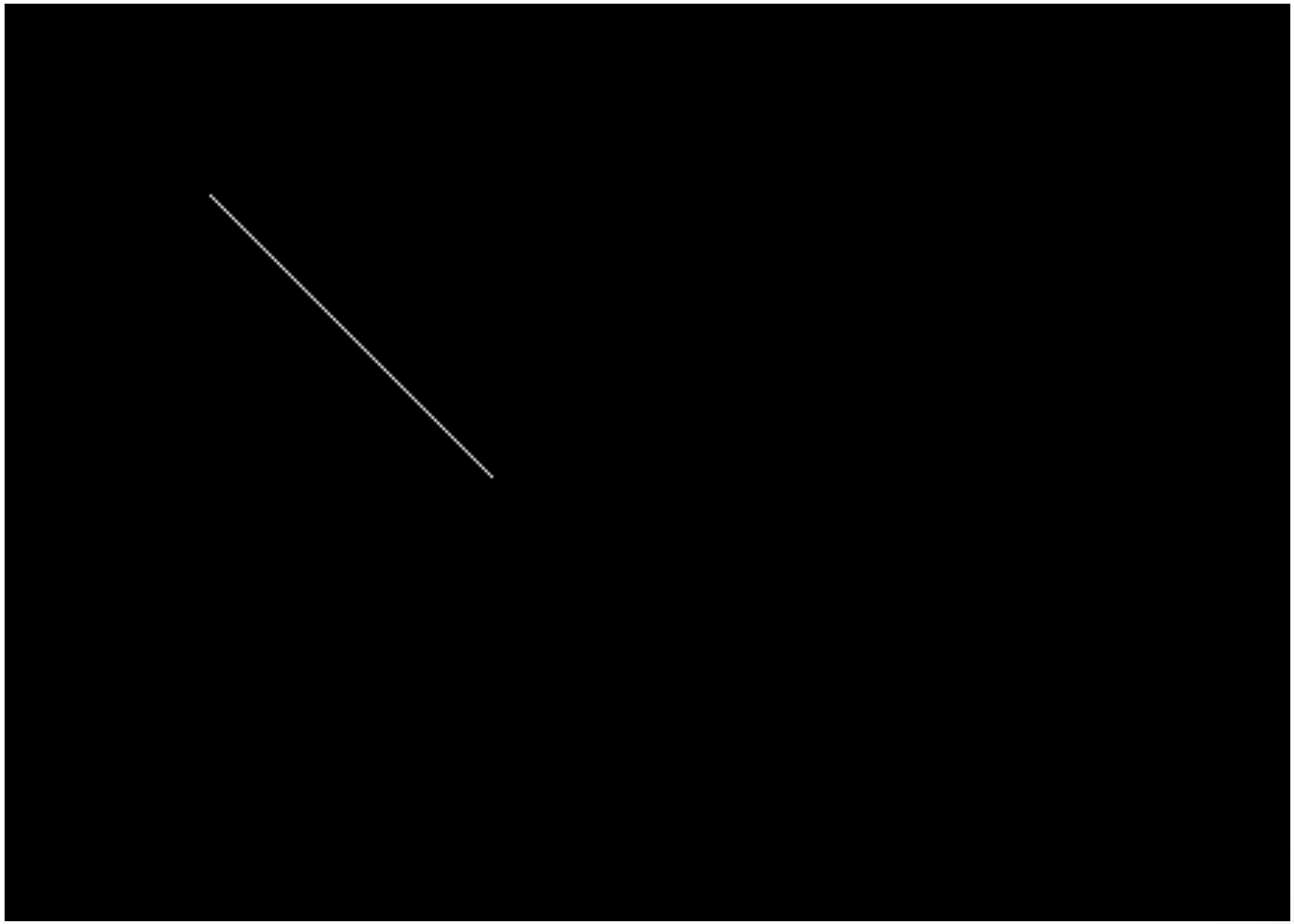
11.3 Graphics Function...

Example 4.

```
/* C graphics program to draw a line */
#include<graphics.h>
#include<conio.h>
void main()
{
   int gd = DETECT, gm;
   /* initialization of graphic mode */
   initgraph(&gd, &gm, "C:\\TC\\BGI");
   line(100,100,200, 200);
   getch();
   closegraph();
}
```

11.3 Graphics Function…

Line() Output



11.3 Graphics Function…

- **arc()**

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.

Declaration:

void arc(int x, int y, int startangle, int endangle, int radius);

11.3 Graphics Function…

arc() example

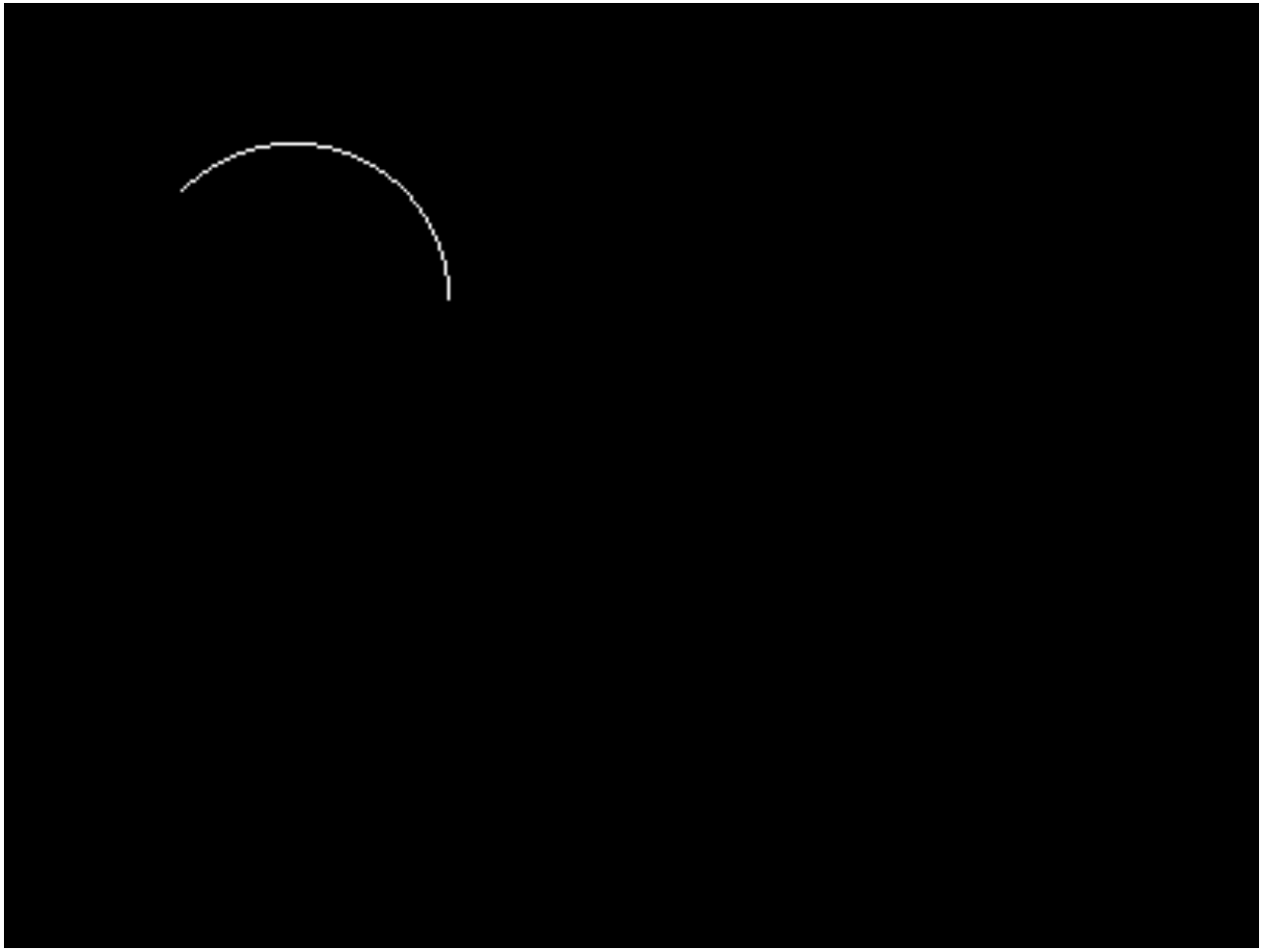
Example 5.

```
#include <graphics.h>
#include <conio.h>
void main()
{
        int gd = DETECT, gm;
        initgraph(&gd, &gm, "C:\\TC\\BGI");
        arc(100, 100, 0, 135, 50);
        getch();
        closegraph();
}
```

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.

11.3 Graphics Function…

arc() example output



11.3 Graphics Function…

- **circle()**

circle:

It draws a circle with radius r and Centre at (x, y)

11.3 Graphics Function…

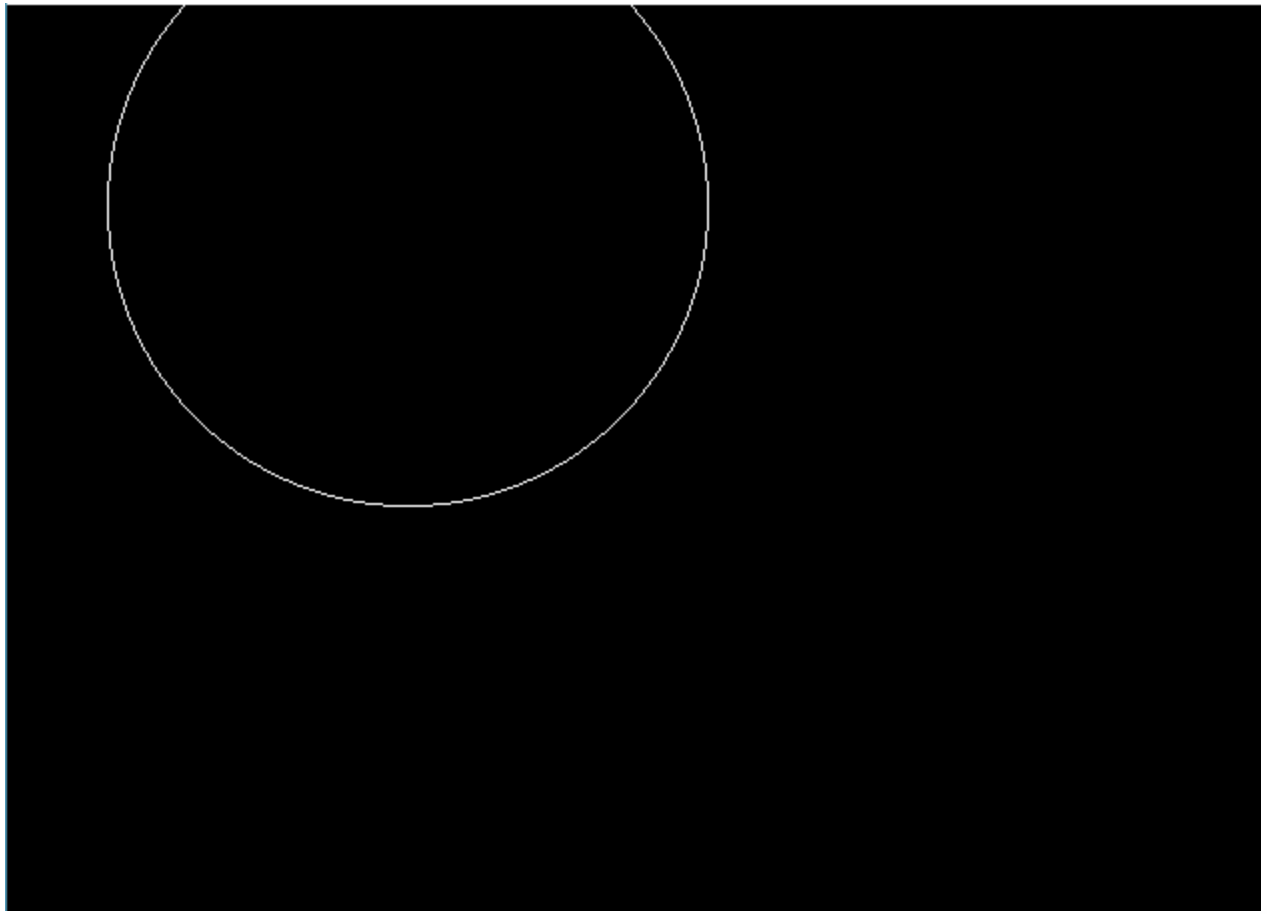
How to Draw a Circle

Example 6.

```
#include<graphics.h>
#include<conio.h>
void main()
{
        int gd=DETECT, gm;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi " );
        circle(200,100,150);
        getch();
        closegraph();
}
```

11.3 Graphics Function…

Output



11.3 Graphics Function…

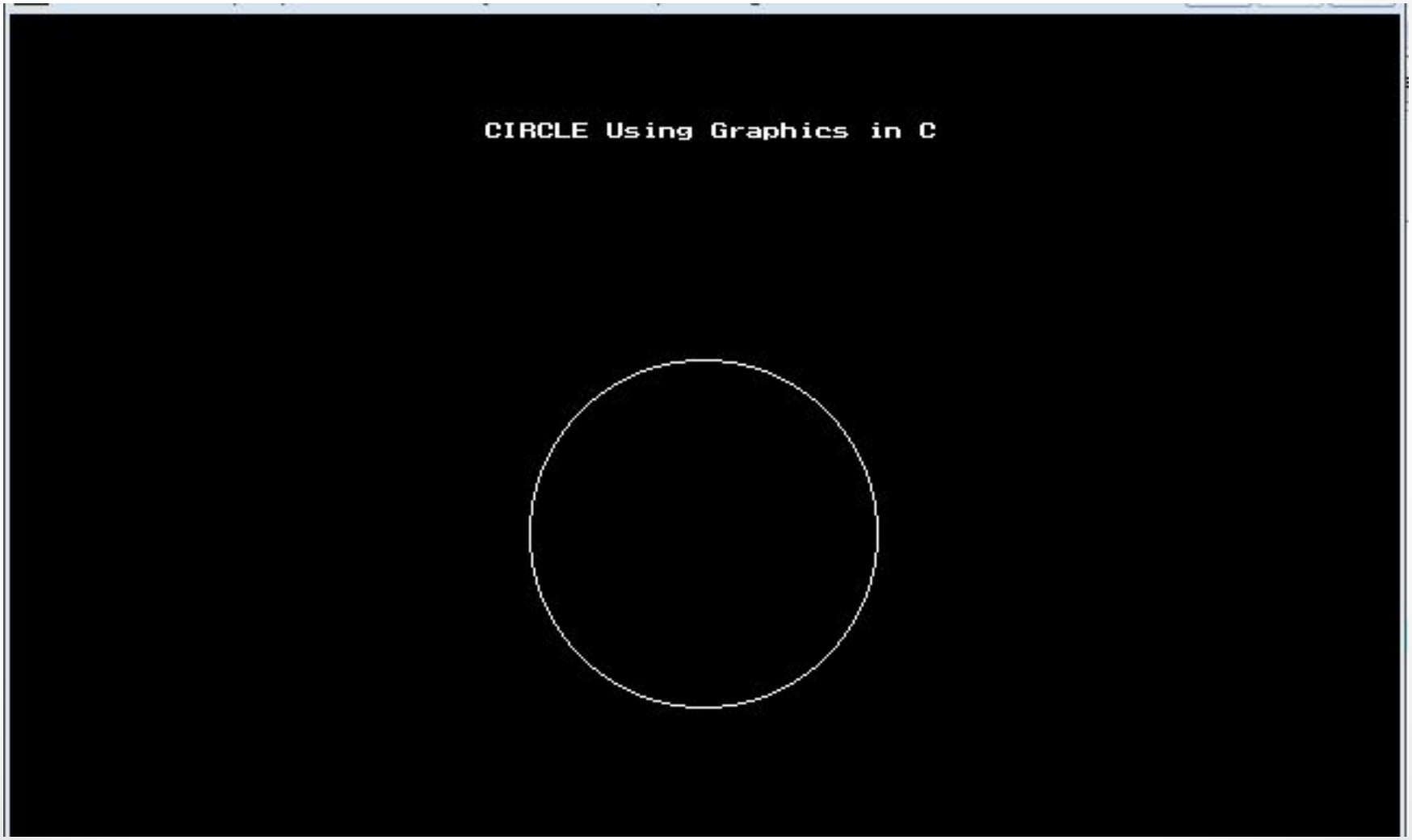
circle()

Example 7.

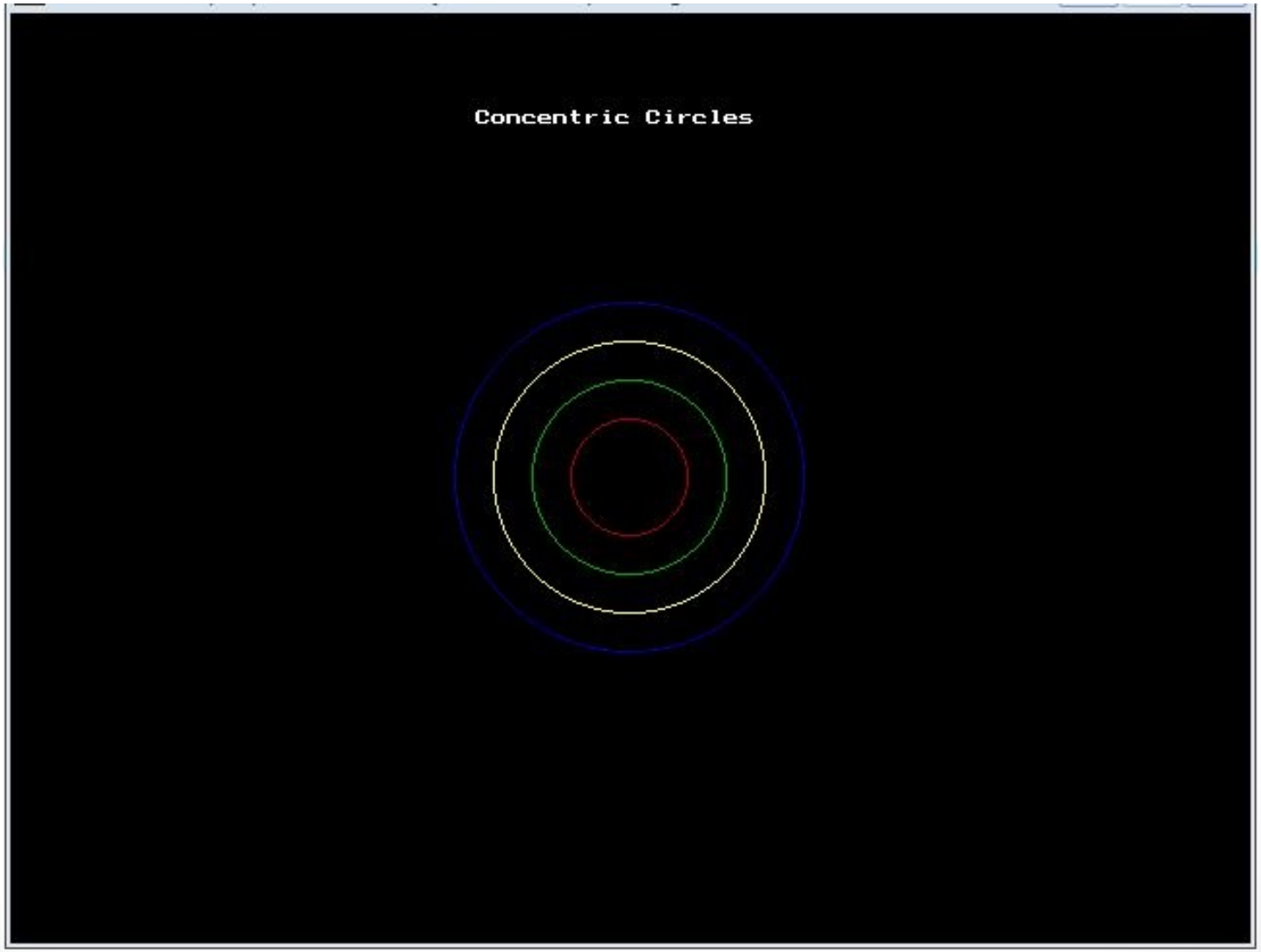
```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>

int main() {
   int gd = DETECT,gm;
   int x ,y ,radius=80;
   initgraph(&gd, &gm, "C:\\TC\\BGI");
   /* Initialize center of circle with center of screen */
   x = getmaxx()/2;
   y = getmaxy()/2;
    outtextxy(x-100, 50, "CIRCLE Using Graphics in C");
   /* Draw circle on screen */
   circle(x, y, radius);
    getch();
   closegraph();
   return 0;
}
```

11.3 Graphics Function…

circle()



//C program to draw concentric circles using graphics



Example 8.

//C program to draw concentric circles using graphics

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

int main(){
   int gd = DETECT,gm;
   int x ,y;
   initgraph(&gd, &gm, "C:\\TC\\BGI");
   /* Initialize center of circle with center of screen */
   x = getmaxx()/2;
   y = getmaxy()/2;
    outtextxy(240, 50, "Concentric Circles");
   /* Draw circles on screen */
   setcolor(RED);
   circle(x, y, 30);
   setcolor(GREEN);
   circle(x, y, 50);
   setcolor(YELLOW);
   circle(x, y, 70);
   setcolor(BLUE);
   circle(x, y, 90);

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

OR,

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

int main(){
   int gd = DETECT,gm;
   int x ,y, n=4;
   initgraph(&gd, &gm, “C:\..\BGI");
   /* Initialize center of circle with center of screen */
   x = getmaxx()/2;
   y = getmaxy()/2;
   outtextxy(240, 50, "Concentric Circles");
   /* Draw circles on screen */
   For(rdi=30;rdi<=90;rdi=rdi+20)
     {      setcolor(n);
            circle(x, y, rdi);
            n++; }
    getch();
   closegraph();
   return 0;
}
```

11.3 Graphics Function…

- **ellipse()** :

In this program, we will draw an eclipse on screen having centre at mid of the screen. We will use **ellipse** functions of graphics.h header file to draw eclipse on screen. Below is the detailed descriptions of ellipse function.

xCenter	: X coordinate of center of eclipse.
yCenter	: Y coordinate of center of eclipse.
startAngle	: Start angle of the eclipse arc.
endAngle	: End angle of the eclipse arc. It will draw eclipse starting form startAngle till endAngle.
xRadius	: Horizontal radius of the eclipse.
yRadius	: Vertical radius of the eclipse.

void ellipse(int xCenter, int yCenter, int startAngle, int endAngle, int xRadius, int yRadius);

11.3 Graphics Function…

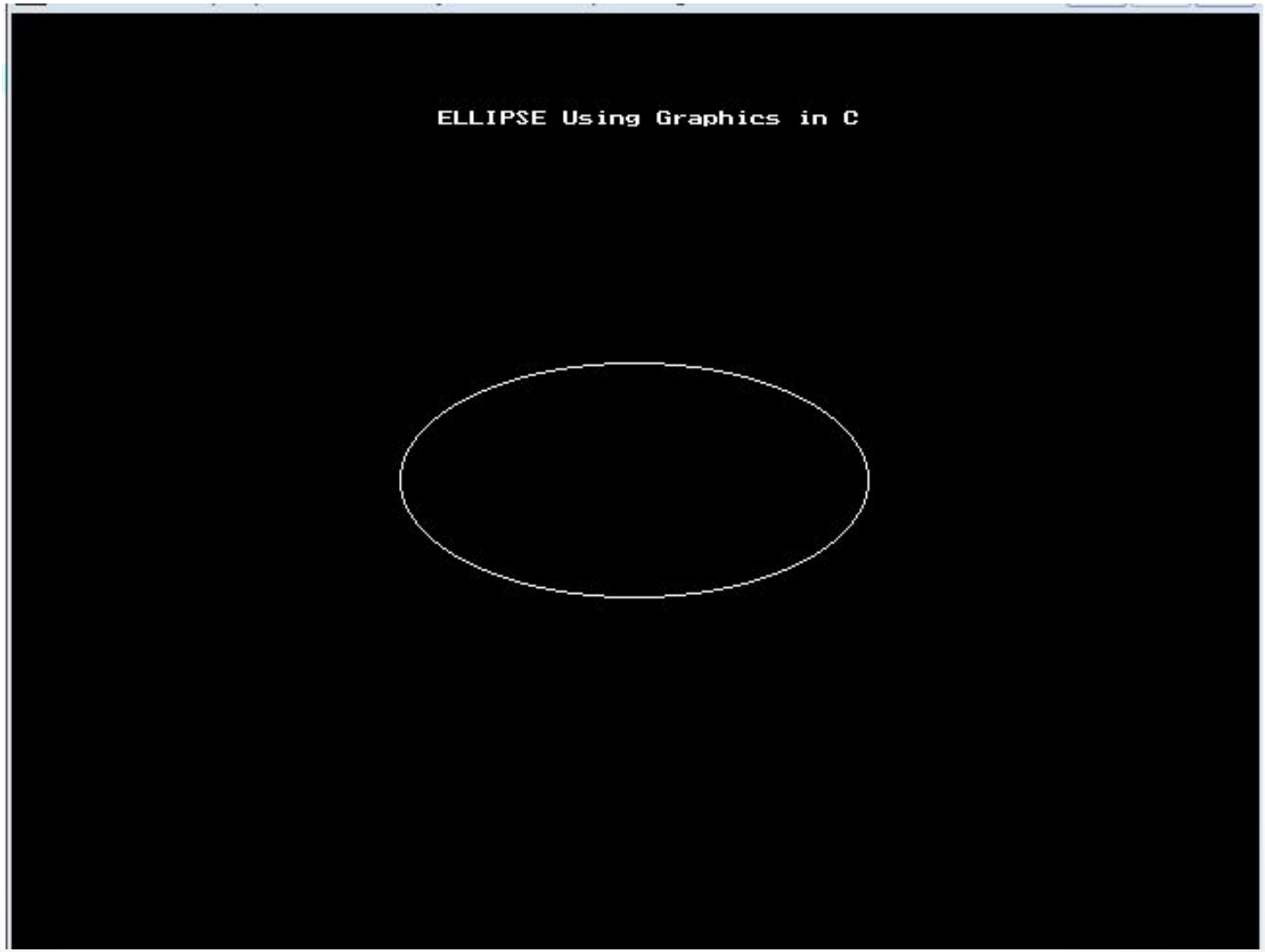
ellipse()

Example 9.

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

int main(){
   int gd = DETECT,gm;
   int x ,y;
   initgraph(&gd, &gm, "X:\\TC\\BGI");
   /* Initialize center of ellipse with center of screen */
   x = getmaxx()/2;
   y = getmaxy()/2;
    outtextxy(x-100, 50, "ELLIPSE Using Graphics in C");
   /* Draw ellipse on screen */
   ellipse(x, y, 0, 360, 120, 60);
    getch();
   closegraph();
   return 0;
}
```

11.3 Graphics Function…

ellipse()



11.3 Graphics Function…

- **floodfill()**

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

floodfill()

```
#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;
}
```

Example 10.

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.

11.3 Graphics Function…

- **getmaxx()** and **getmaxy()**

getmaxx :

It returns the maximum X coordinate in current graphics mode and driver.

getmaxy()

It returns the maximum Y coordinate in current graphics mode and driver.

11.3 Graphics Function…

- **outtextxy()**

Outtextxy :

It displays a string at a particular point (x,y) on screen.

//Drawing concentric circles

Example 11.

```
#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;
}
```

// C graphics program moving car

Example 12.

```
#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;
}
```

C Program to Draw a Rectangle and Bar Using C Graphics

void rectangle(int xTopLeft, int yTopLeft, int xBottomRight, int yBottomRight);

void bar(int xTopLeft, int yTopLeft, int xBottomRight, int yBottomRight);

C Program to Draw a Rectangle and Bar Using C Graphics

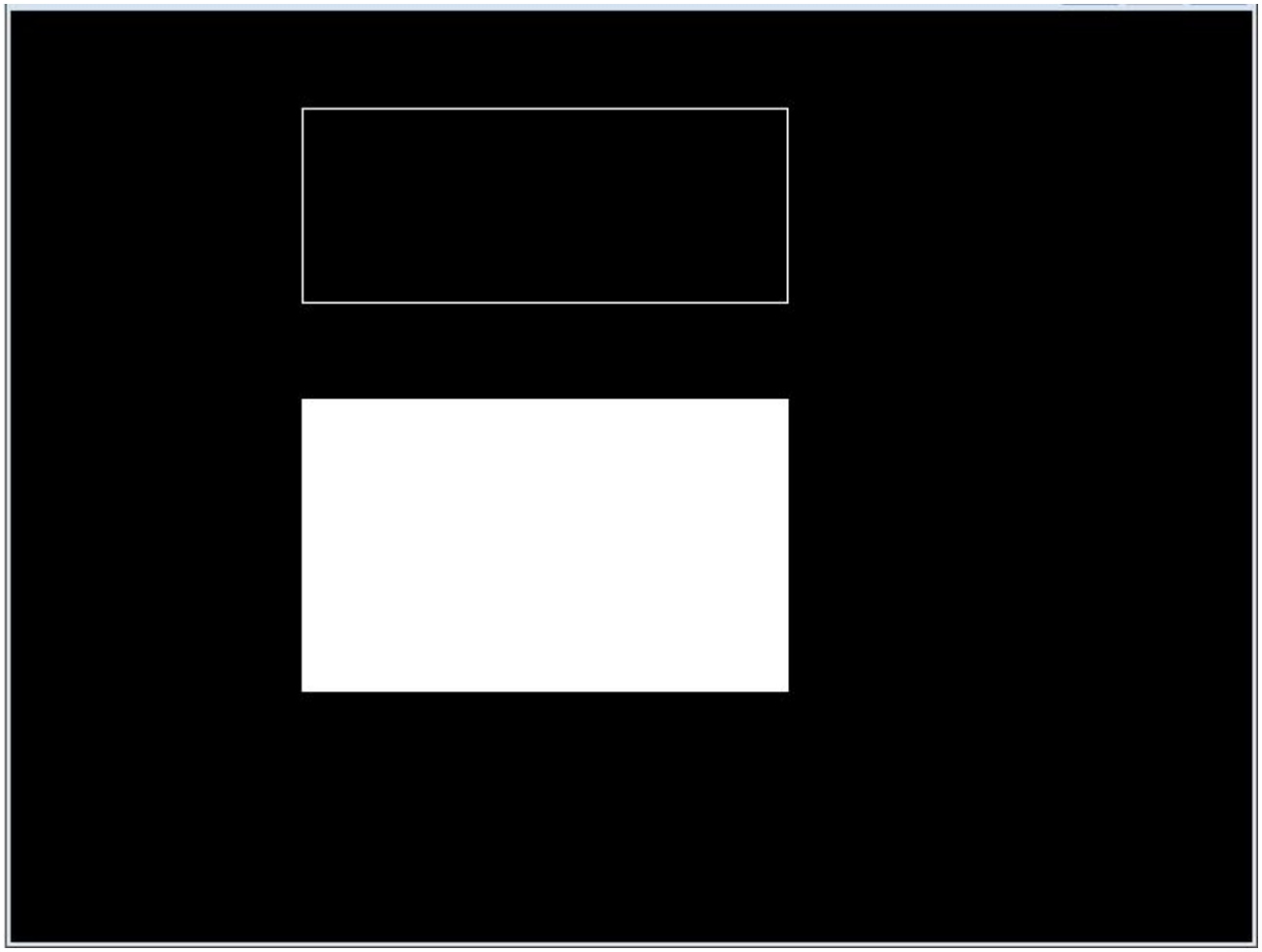
Function Argument	Description
xTopLeft	X coordinate of top left corner.
yTopLeft	Y coordinate of top left corner.
xBottomRight	X coordinate of bottom right corner.
yBottomRight	Y coordinate of bottom right corner.

//C program to draw rectangle and bar using graphics

Example 13.

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

int main(){
   int gd = DETECT,gm;
   initgraph(&gd, &gm, "C:\\TC\\BGI");

   /* Draw rectangle on screen */
   rectangle(150, 50, 400, 150);

   /* Draw Bar on screen */
   bar(150, 200, 400, 350);

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

output



C Program to Set the Fill style Using C Graphics

we will draw a bar graph on screen. Here, we will use **line**, **setfillstyle** and **bar** functions of **graphics.h** header file to draw horizontal and vertical axis and bars on screen.

void setfillstyle(int pattern, int color);

Some set fill style function are :

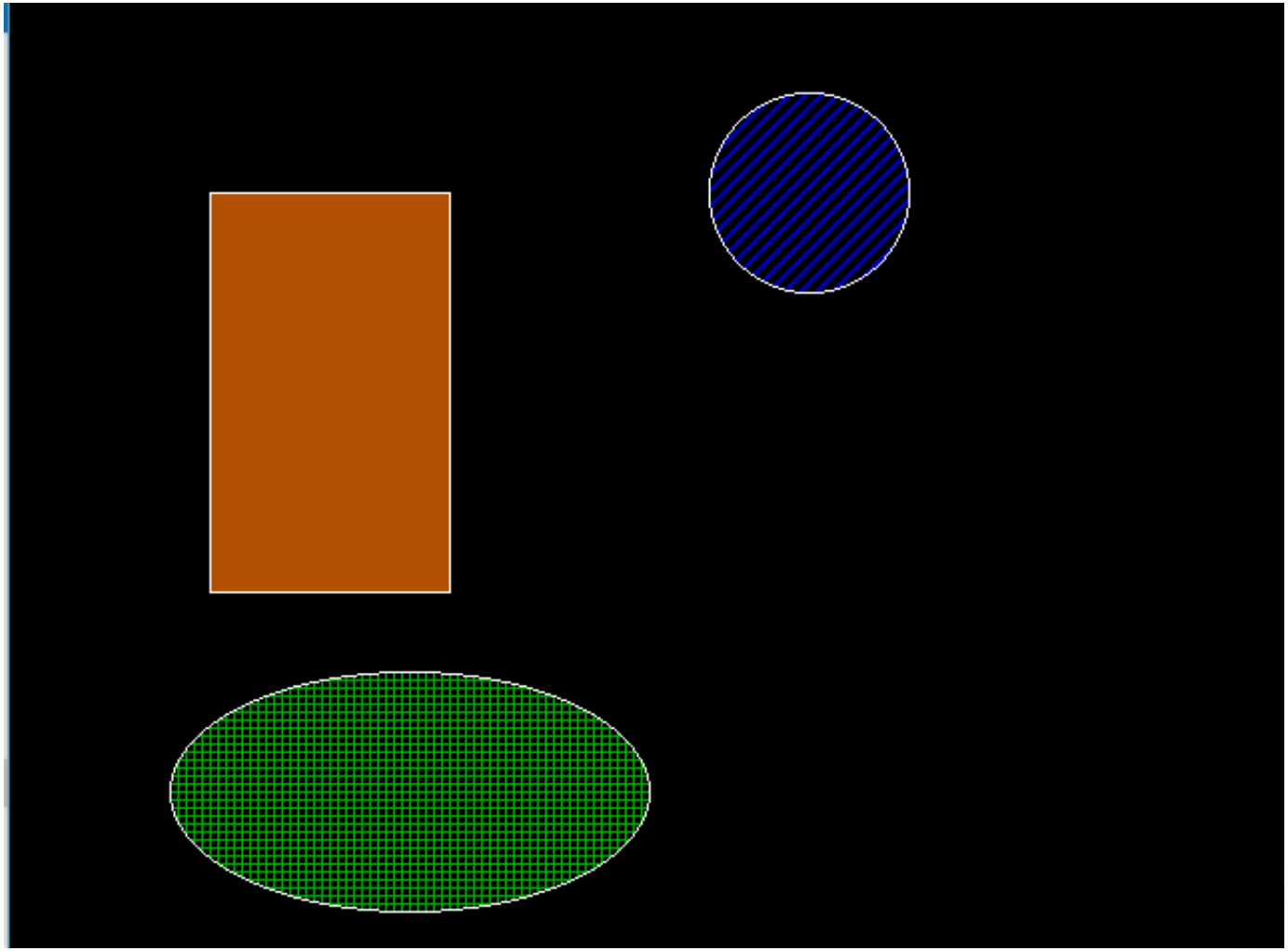
1)EMPTY_FILL,
2)SOLID_FILL ,
3)LINE_FILL,
4)LTSLASH_FILL ,
5)SLASH_FILL,
6)BKSLASH_FILL,
7)LTBKSLASH_FILL,
8) HATCH_FILL,
9) XHATCH_FILL,
10) INTERLEAVE_FILL,
11) WIDE_DOT_FILL,
12) CLOSE_DOT_FILL,
13) USER_FILL

C Program to Set the Fill style Using C Graphics

Example 14.

```
#include<graphics.h>
#include<conio.h>
 int main()
{
 int gd = DETECT,gm;
 initgraph(&gd, &gm, "X:\\TC\\BGI");
            setcolor(WHITE);
            rectangle(100,100,220,300);
            setfillstyle(SOLID_FILL, BROWN);
            floodfill(101, 101, WHITE);

            circle(400,100,50);
            setfillstyle(SLASH_FILL, BLUE);
            floodfill(400, 100, WHITE);

            ellipse(200,400, 0, 360, 120, 60);
            setfillstyle(HATCH_FILL, GREEN);
            floodfill(200, 400, WHITE);

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

output

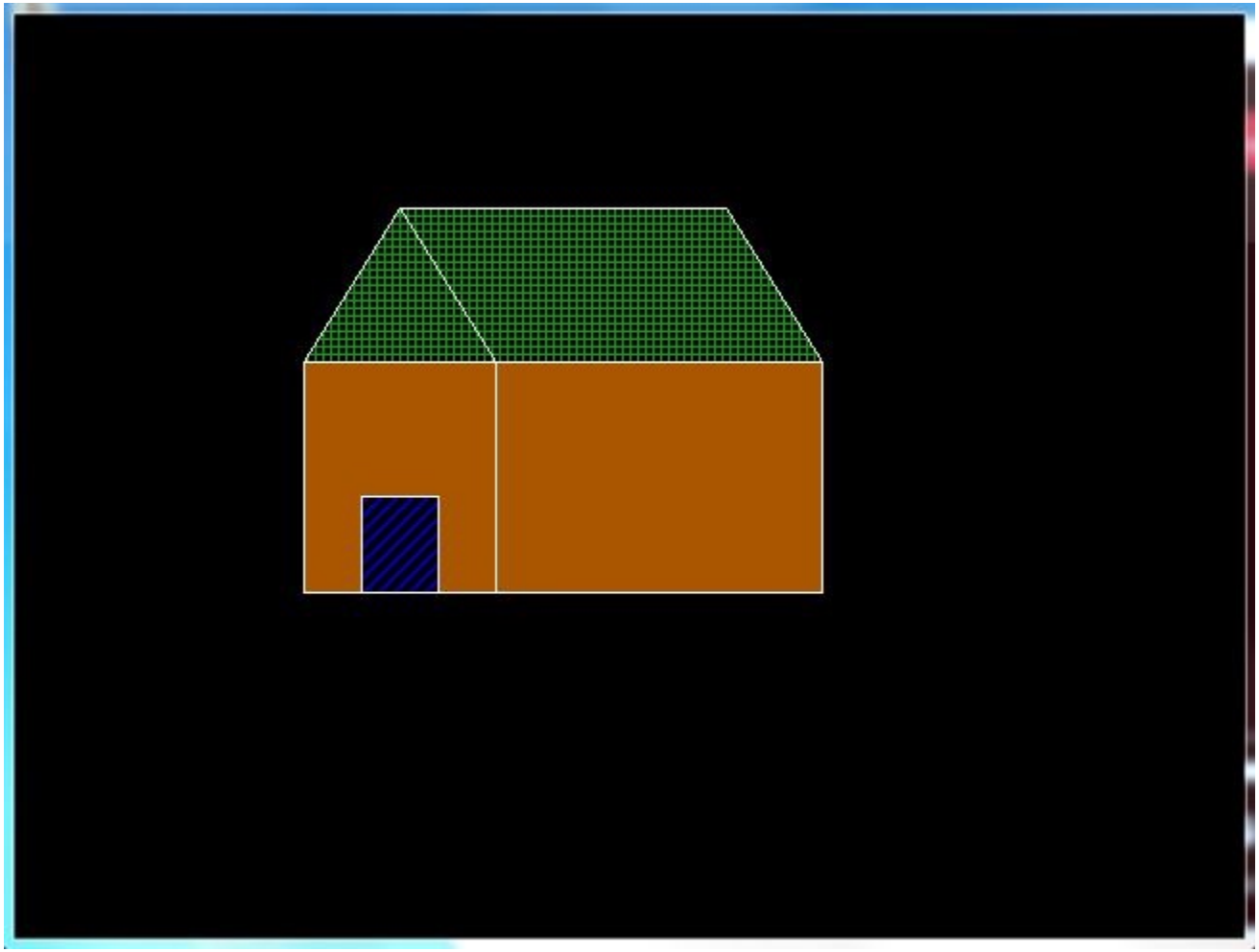


//C program to draw a hut and color it using graphics

```
#include<graphics.h>
#include<conio.h>
 int main() {
 int gd = DETECT,gm;
 initgraph(&gd, &gm, "X:\\TC\\BGI");
   /* Draw Hut */
                setcolor(WHITE);
               rectangle(150,180,250,300);
               rectangle(250,180,420,300);
               rectangle(180,250,220,300);
               line(200,100,150,180);
               line(200,100,250,180);
               line(200,100,370,100);
               line(370,100,420,180);
   /* Fill colours */
               setfillstyle(SOLID_FILL, BROWN);
               floodfill(152, 182, WHITE);
               floodfill(252, 182, WHITE);
               setfillstyle(SLASH_FILL, BLUE);
               floodfill(182, 252, WHITE);
               setfillstyle(HATCH_FILL, GREEN);
               floodfill(200, 105, WHITE);
               floodfill(210, 105, WHITE);
               getch();    closegraph();    return 0;
}
```

Example 15.

Output



WAP to draw a red circle and show the text “Fig.circle having radius 50” and using GOTHIC_FONT with HORIZ_DIR style and size 2.

```
#include<graphics.h>
#include<conio.h>
void main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"");
    setcolor(RED);
    circle(200,200,50);
    outtextxy(200,100,".");
    outtextxy(175,110,"(200,100)");
    settextstyle(GOTHIC_FONT, HORIZ_DIR,2);
    outtextxy(50,175,"Fig. circle having radius 50");
    getch();
    closegraph();
}
```

Thank You

Finished

Unit 11