Program 2

Aim: Write a program to draw/create a car (without tyres) using Mid-Point Algorithm/Bresenham Algorithm.

Theory:

Bresenham's line algorithm is an algorithm that determines the points of an n-dimensional raster that should be selected in order to form a close approximation to a straight line between two points. It is commonly used to draw line primitives in a bitmap image, as it uses only integer addition, subtraction and bit shifting, all of which are very cheap operations in standard computer architectures. It is an incremental error algorithm. It is one of the earliest algorithms developed in the field of computer graphics.

Algorithm:

Step 1 − Input the two end-points of line, storing the left end-point in (X0,Y0).

Step 2 − Plot the point (X0,Y0).

Step 3 − Calculate the constants dx, dy, 2dy, and (2dy – 2dx) and get the first value for the decision parameter as −

$$\mathbf{p0 = 2dy - dx}$$

Step 4 − At each Xk along the line, starting at k = 0, perform the following -

If pkpk < 0, the next point to plot is (Xk+1,Yk) and

$$\mathbf{Pk + 1 = Pk + 2dy}$$

Otherwise, (Xk,Yk+1)

$$\mathbf{Pk + 1 = Pk + 2dy - 2dx}$$

Step 5 − Repeat Step 4 (dx – 1) times.

Code:

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

void midPointLineAlgo(int X1, int Y1, int X2, int Y2){

    int dx, dy, lastX, lastY, i, nextX, nextY;
    int p, x, y, t, d;
    if(X2<X1){
        t = X1;
        X1 = X2;
        X2 = t;
        t = Y1;
        Y1 = Y2;
        Y2 = t;
    }
    dx = X2 - X1;
    dy = Y2 - Y1;

    if(dx == 0)
        dx = 1;

    lastX = X1;
    lastY = Y1;

    if(abs(dy)>abs(dx)){
        d = 2 * abs(dx) - abs(dy);
        if(dy>0 && dx>0){
            for (i=1; i<=abs(dy); i++){
                nextY = lastY + 1;
                if (d>0){
                    nextX = lastX + 1;
                    d = d + 2 * abs(dx) - 2 * abs(dy);
                }else{
                    nextX = lastX;
                    d = d + 2 * abs(dx);
                }
                putpixel(lastX, lastY, WHITE);
                lastX = nextX;
                lastY = nextY;
            }
        }else{
            for(i=1; i<=abs(dy); i++){
                nextY = lastY - 1;
                if(d<0){
                    nextX = lastX;
                    d = d + 2 * abs(dx);
                }else{
                    nextX = lastX + 1;
                    d = d + 2 * abs(dx) - 2 * abs(dy);
                }
                putpixel(lastX, lastY, WHITE);
                lastX = nextX;
                lastY = nextY;
            }
        }
    }else{
        d = 2 * abs(dy) - abs(dx);
        if(dy>0 && dx>0){
            for(i=1; i<=abs(dx); i++){
                nextX = lastX + 1;
                if(d<0){
                    nextY = lastY;
                    d = d + 2 * abs(dy);
                }else{
                    nextY = lastY + 1;
                    d = d + 2 * abs(dy) - 2 * abs(dx);
                }
                putpixel(lastX, lastY, WHITE);
                lastX = nextX;
                lastY = nextY;
            }
        }else{
            for(i=1; i<=abs(dx); i++){
                nextX = lastX + 1;
                if(d>=0){
                    nextY = lastY - 1;
                    d = d + 2 * abs(dy) - 2 * abs(dx);
                }else{
                    nextY = lastY;
                    d = d + 2 * abs(dy);
                }
                putpixel(lastX, lastY, WHITE);
                lastX = nextX;
                lastY = nextY;
            }
        }
    }
}

int main(){
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
    midPointLineAlgo(200,200,400,200);
    midPointLineAlgo(400,200,500,300);
    midPointLineAlgo(200,200,100,300);

    midPointLineAlgo(500,300,570,300);
    midPointLineAlgo(570,300,570,380);
    midPointLineAlgo(100,300,30,300);
    midPointLineAlgo(30,380,570,380);
    midPointLineAlgo(30,300,30,380);

    midPointLineAlgo(220,220,280,220);
    midPointLineAlgo(280,220,280,280);
    midPointLineAlgo(220,220,220,280);
    midPointLineAlgo(280,280,220,280);

    midPointLineAlgo(320,220,380,220);
    midPointLineAlgo(320,220,320,280);
    midPointLineAlgo(320,280,380,280);
    midPointLineAlgo(380,280,380,220);

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

Output: