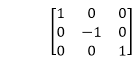
Reflection:

It is a transformation which produces a mirror image of an object. The mirror image can be either about x-axis or y-axis. The object is rotated by180°.

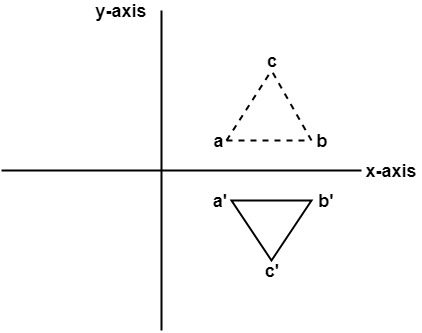
Types of Reflection:

1. Reflection about the x-axis
2. Reflection about the y-axis
3. Reflection about an axis perpendicular to xy plane and passing through the origin
4. Reflection about line y=x

**1. Reflection about x-axis:** The object can be reflected about x-axis with the help of the following matrix



In this transformation value of x will remain same whereas the value of y will become negative. Following figures shows the reflection of the object axis. The object will lie another side of the x-axis.

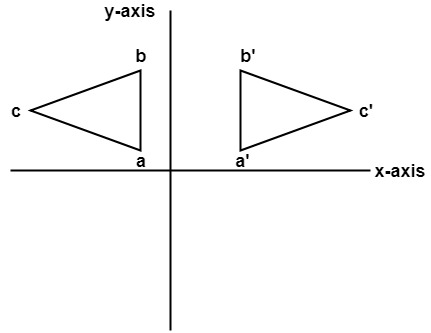


**2. Reflection about y-axis:** The object can be reflected about y-axis with the help of following transformation matrix

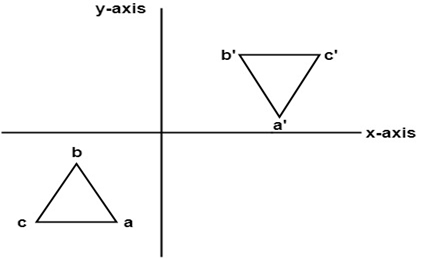
Reflection

Here the values of x will be reversed, whereas the value of y will remain the same. The object will lie another side of the y-axis.

The following figure shows the reflection about the y-axis

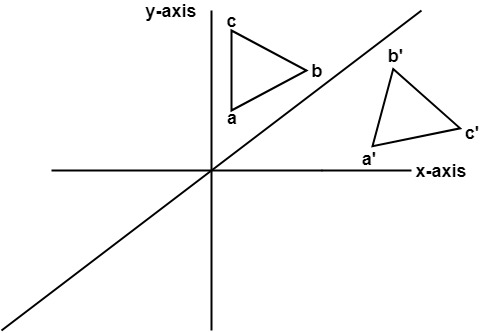


**3. Reflection about an axis perpendicular to xy plane and passing through origin:**  
In the matrix of this transformation is given below

Reflection  


In this value of x and y both will be reversed. This is also called as half revolution about the origin.

**4. Reflection about line y=x:** The object may be reflected about line y = x with the help of following transformation matrix

Reflection  


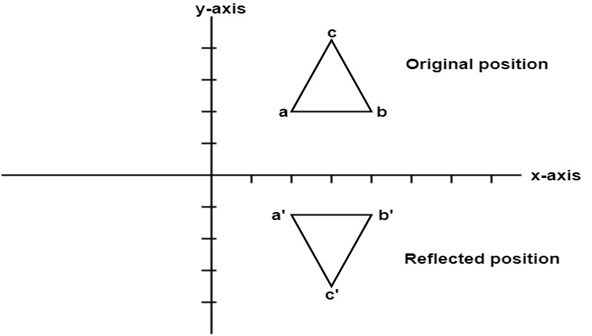
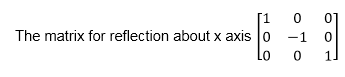
First of all, the object is rotated at 45°. The direction of rotation is clockwise. After it reflection is done concerning x-axis. The last step is the rotation of y=x back to its original position that is counterclockwise at 45°.

**Example:** A triangle ABC is given. The coordinates of A, B, C are given as

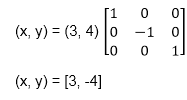
                    A (3 4)  
                    B (6 4)  
                    C (4 8)

Find reflected position of triangle i.e., to the x-axis.

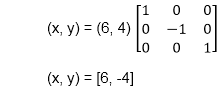
**Solution:**

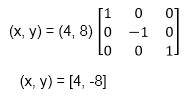
The a point coordinates after reflection



The b point coordinates after reflection



The coordinate of point c after reflection



a (3, 4) becomes a1 (3, -4)  
b (6, 4) becomes b1 (6, -4)  
c (4, 8) becomes c1 (4, -8)

Program to perform Mirror Reflection about a line:

1. #include <iostream.h>
2. #include <conio.h>
3. #include <graphics.h>
4. #include <math.h>
5. #include <stdlib.h>
6. #define pi 3.14
7. **class** arc
8. {
9. **float** x[10],y[10],theta,ref[10][10],ang;
10. **float** p[10][10],p1[10][10],x1[10],y1[10],xm,ym;
11. **int** i,k,j,n;
12. **public**:
13. **void** get();
14. **void** cal ();
15. **void** map ();
16. **void** graph ();
17. **void** plot ();
18. **void** plot1();
19. };
20. **void** arc::get ()
21. {
22. cout<<"\n ENTER ANGLE OF LINE INCLINATION AND Y INTERCEPT";
23. cin>> ang >> b;
24. cout <<"\n ENTER NO OF VERTICES";
25. cin >> n;
26. cout <<"\n ENTER";
27. **for** (i=0; i<n; i++)
28. {
29. cout<<"\n x["<<i<<"] and y["<<i<<"]";
30. }
31. theta =(ang \* pi)/ 180;
32. ref [0] [0] = cos (2 \* theta);
33. ref [0] [1] = sin (2 \* theta);
34. ref [0] [2] = -b \*sin (2 \* theta);
35. ref [1] [0] = sin (2 \* theta);
36. ref [1] [1] = -cos (2 \* theta);
37. ref [1] [2] = b \* (cos (2 \* theta)+1);
38. ref [2] [0]=0;
39. ref [2] [1]=0;
40. ref [2] [2] = 1;
41. }
42. **void** arc :: cal ()
43. {
44. **for** (i=0; i < n; i++)
45. {
46. p[0] [i] = x [i];
47. p [1] [i] = y [i];
48. p [2] [i] = 1;
49. }
50. **for** (i=0; i<3;i++)
51. {
52. **for** (j=0; j<n; j++)
53. {
54. p1 [i] [j]=0;
55. **for** (k=0;k<3; k++)
56. }
57. p1 [i] [j] + = ref [i] [k] \* p [k] [j];
58. }
59. **for** (i=0; i<n; i++)
60. {
61. x1 [i]=p1[0] [i];
62. y1 [i] = p1 [1] [i];
63. }
64. }
65. **void** arc :: map ()
66. {
67. **int** gd = DETECT,gm;
68. initgraph (&gd, &gm, " ");
69. **int** errorcode = graphresult ();
70. /\* an error occurred \*/
71. **if** (errorcode ! = grOK)
72. {
73. printf ("Graphics error: %s \n", grapherrormsg (errorcode));
74. printf ("Press any key to halt:");
75. getch ();
76. exit (1); /\* terminate with an error code \*/
77. }
78. }
79. **void** arc :: graph ()
80. {
81. xm=getmaxx ()/2;
82. ym=getmaxy ()/2;
83. line (xm, 0, xmm 2\*ym);
84. }
85. **void** arc :: plot 1 ()
86. {
87. **for** (i=0; i <n-1; i++)
88. {
89. circle (x1[i]+xm, (-y1[i]+ym), 2);
90. line (x1[i]+xm, (-y1[i]+ym), x1[i+1]+xm, (-y1[i+1]+ym));
91. }
92. line (x1[n-1)+xm, (-y1[n-1]+ym), x1[0]+xm, (-y1[0]+ym));
93. getch();
94. }
95. **void** arc :: plot ()
96. {
97. **for** (i=0; i <n-1; i++)
98. {
99. circle (x1[i]+xm, (-y1[i]+ym, 2);
100. line (x1[i]+xm, (-y1[i]+ym), x[i+1]+xm, (-y1[i+1]+ym));
101. }
102. line (x[n-1]+xm, (-y1[n-1]+ym), x[0]+xm, (-y[0]+ym));
103. getch();
104. }
105. **void** main ()
106. {
107. **class** arc a;
108. clrscr();
109. a.map();
110. a.graph();
111. a.get();
112. a.cal();
113. a.plot();
114. a.plot1();
115. getch();
116. }

**Output:**

