```
/* Dealing with vector and vector of vectors of multi-dimension points */

// vector of 2-D floating points
vector<Point2f> ViewCorners, Corners;
// vector of 3-D floating points
vector<Point3f> RealCorners;
// vector of characters
vector<uchar> Status;
// vector of vectors of 2-D floating points
vector<vector<Point2f>> Point_2D;
// vector of vectors of 3-D floating points
vector<vector<Point3f>> Point_3D;
// 3-D floating point
Point3f 3D_Point

Point_2D->clear()                       // clear vectors
Point_3D->clear()                       // clear vectors

ViewCorners->push_back(Corners(i));     // push one 2D corner point (indexed by
    i) to vector ViewCorners
3D_Point.x = 100;                       // assign floating point value to x of 3D
    point
3D_Point.y = 200;                       // assign floating point value to y of 3D
    point
3D_Point.z = 0.0f;                      // assign floating point value to z of 3D
    point
RealCorners->push-back(3D_Point);       // push one 3D corner point to vector
    RealCorner

Point_2D->push_back(ViewCorners);       // push vector of 2D points to a vector
    of vector Point_2D
Point_3D->push_back(RealCorners);       // push vector of 3D points to a vector
    of vector Point_3D
ViewCorners.clear();                    // clear vector
RealCorners.clear();                    // celar vector

Corners.at(i).x = 10;                   // access individual verctor of Point2f
Corners.at(i).y = 10;

/* Dealing with vector and vector of vectors of matrices */

// Matrix for a predefined type (unsigned char) and dimension (480 rows and 640
    columns)
Mat Buffer(480, 640, CV_8UC1);
// Matrix of 3 by 3 double
Mat CamMatrix = Mat(3,3,CV_64FC1);
// Matrix of 5 by 1 double
Mat CamDistort = Mat(5,1,CV_64FC1);
// Matrix without predefined type and dimension
Mat PreviousPoint;
// Vector of matrices
vector<Mat> RVect;
vector<Mat> TVect;

CamMatrix.at<double>(i, j) = 1.5f;      // access (i, j) component of a matrix;
cout << CamMatrix.row(i) << "\n";       // print/access the entire row of a
```

```
    matrix
cout << CamMatrix.col(i) << "\n";       // print/access the entire col of a
    matrix
PreviousPoint = Mat(Point_2D.at(i));    // convert one of the vectors (indexed by
    i) of a vector of vectors 2D points
PreviousPoint.at<Point2f>(i).x = 100;       // access x of point i of a 2D point
    matrix
PreviousPoint.at<Point2f>(i).y = 100;       // access y of point i of a 2D point
    matrix

/* 3D vector for a line ax + by + c = 0;
vector<Vec3f> Line
Line[i][0] : a      // vector is indexed by i
Line[i][1] : b
Line[i][2] : c
```