```
/*
 K-means
 */
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;
typedef struct  //point struct
{
    double x;  //attribute1
    double y;  //attribute2
}Point;

int main()
{
    vector<Point> vecPoint;   //all points
    vector<Point> vecCenter;  //centorids
    vector<vector<Point> > vecCluster;  //cluster1, 2, 3
    double
tempM[8][2]={1.0,1.0,1.0,2.0,2.0,1.0,2.0,3.0,3.0,3.0,4.0,5.0,5.0,4.0,6.0,5.0}
;//8 points
    Point temp;
    int i;
    for(i=0;i<8;i++) //initialize each point
    {
        temp.x=tempM[i][0];
        temp.y=tempM[i][1];
        vecPoint.push_back(temp);
    }
    //initialize centorids
    temp.x=2.0;
    temp.y=3.0;
    vecCenter.push_back(temp);
    temp.x=3.0;
    temp.y=3.0;
    vecCenter.push_back(temp);
    temp.x=5.0;
    temp.y=4.0;
    vecCenter.push_back(temp);
    double distance; //distance
    double mindistance;
    unsigned int flag;  //
    unsigned int round=1;  //iteration;
    double errorSum1;  //within cluster sum of squared error1
    double errorSum2;  //within cluster sum of squared error2
    double errorSum3;  //between cluster sum of squared error3
    double errorSum4;  //between cluster sum of squared error4
    vector<Point> tempvec;
    //double distance1,distance2,distance3;
    for(i=0;i<3;i++)   //initialize Cluster vector
    {
        vecCluster.push_back(tempvec);
    }
    do
    {
```

```
        for(i=0;i<vecCluster.size();i++)  //clear vector of each cluster
        {
            vecCluster[i].clear();
        }
        for(i=0;i<8;i++)    //sort data
        {
            mindistance=sqrt(pow(vecPoint[i].x-
vecCenter[0].x,2.0)+pow(vecPoint[i].y-vecCenter[0].y,2.0));
            flag=0;
            for(int k=1;k<3;k++)   //compare distance with centorids
            {
                distance=sqrt(pow(vecPoint[i].x-
vecCenter[k].x,2.0)+pow(vecPoint[i].y-vecCenter[k].y,2.0));
                if(distance<mindistance)  //closer to centorid
                {
                    flag=k;
                    mindistance=distance;
                }
            }
            vecCluster[flag].push_back(vecPoint[i]);  //sort data to cluster
        }// end of for(i=0;i<8;i++)

        cout<<"-------------"<<round<<" result:-------------"<<endl;
        for(i=0;i<vecCluster.size();i++)  //output clusters
        {
            cout<<"cluster"<<i+1<<": ";
            for(int j=0;j<vecCluster[i].size();j++)
            {
                cout<<"("<<vecCluster[i][j].x<<","<<vecCluster[i][j].y<<") ";
            }
            cout<<"centorid
selected:("<<vecCenter[i].x<<","<<vecCenter[i].y<<")";
            cout<<endl;
        }

        if(round==1)
        {
            //compute initial WSS
            errorSum1=0;
            errorSum3=0;
            for(int k=0;k<3;k++)
            {
                for(i=0;i<vecCluster[k].size();i++)
                {
                    errorSum1+=pow(vecCenter[k].x-
vecCluster[k][i].x,2.0)+pow(vecCenter[k].y-vecCluster[k][i].y,2.0);
                }
                //the centroid of all data points is (3,3)
                errorSum3+=vecCluster[k].size()*(pow(vecCenter[k].x-3,
2.0)+pow(vecCenter[k].y-3, 2.0));
            }
            cout<<"WSS = "<<errorSum1<<endl;

        }
        else
```

```
        {
            errorSum1=0;

            errorSum3=0;
            for(int k=0;k<3;k++)
            {
                for(i=0;i<vecCluster[k].size();i++)
                {
                    errorSum1+=pow(vecCenter[k].x-
vecCluster[k][i].x,2.0)+pow(vecCenter[k].y-vecCluster[k][i].y,2.0);
                }
                //the centroid of all data points is (3,3)
                errorSum3+=vecCluster[k].size()*(pow(vecCenter[k].x-3,
2.0)+pow(vecCenter[k].y-3, 2.0));
            }
            cout<<"WSS = "<<errorSum1<<endl;
            errorSum1=errorSum2;  //record last WSS
        }

        cout<<"BSS = "<<errorSum3<<endl;


        round++;  //iteration+1
        double sum_x,sum_y;
        vecCenter.clear();  //clear centorid vector
        int k;
        for(k=0;k<3;k++)   //recompute centorids
        {
            sum_x=0;
            sum_y=0;
            for(i=0;i<vecCluster[k].size();i++)
            {
                sum_x+=vecCluster[k][i].x;
                sum_y+=vecCluster[k][i].y;
            }
            temp.x=sum_x/vecCluster[k].size();
            temp.y=sum_y/vecCluster[k].size();
            vecCenter.push_back(temp);  //
        }
        errorSum2=0;  //compute new WSS

        for(k=0;k<3;k++)
        {
            for(i=0;i<vecCluster[k].size();i++)
            {
                errorSum2+=pow(vecCenter[k].x-
vecCluster[k][i].x,2.0)+pow(vecCenter[k].y-vecCluster[k][i].y,2.0);
            }
        }
        /*
         distance1=fabs(vecCenter[0].x-vecCenter[3].x)+fabs(vecCenter[0].y-
vecCenter[3].y);
         distance2=fabs(vecCenter[1].x-vecCenter[4].x)+fabs(vecCenter[1].y-
vecCenter[4].y);
         distance3=fabs(vecCenter[2].x-vecCenter[5].x)+fabs(vecCenter[2].y-
```

```
vecCenter[5].y);
         vecCenter.erase(vecCenter.begin(),vecCenter.begin()+3);
         */
    }while(fabs(errorSum2-errorSum1)>0.0000000001);   //see if WSS is being
stable
    //}while(distance1||distance2||distance3);
    cout<<endl<<">>>WSS is being stable, Stop"<<endl;
    cout<<"------------Final Clustering Result:-------------"<<endl;
    for(i=0;i<vecCluster.size();i++)  //output final clustering result
    {
        cout<<"cluster"<<i+1<<": ";
        for(int j=0;j<vecCluster[i].size();j++)
        {
            cout<<"("<<vecCluster[i][j].x<<","<<vecCluster[i][j].y<<") ";
        }
        cout<<endl;
    }
    return 0;
}
```

Result screenshot:

```
-------------1 result:-------------
cluster1: (1,1) (1,2) (2,1) (2,3) centorid selected:(2,3)
cluster2: (3,3) centorid selected:(3,3)
cluster3: (4,5) (5,4) (6,5) centorid selected:(5,4)
WSS = 15
BSS = 19
-------------2 result:-------------
cluster1: (1,1) (1,2) (2,1) centorid selected:(1.5,1.75)
cluster2: (2,3) (3,3) centorid selected:(3,3)
cluster3: (4,5) (5,4) (6,5) centorid selected:(5,4.66667)
WSS = 5.60417
BSS = 31.7708
-------------3 result:-------------
cluster1: (1,1) (1,2) (2,1) centorid selected:(1.33333,1.33333)
cluster2: (2,3) (3,3) centorid selected:(2.5,3)
cluster3: (4,5) (5,4) (6,5) centorid selected:(5,4.66667)
WSS = 4.5
BSS = 37.5

>>>WSS is being stable, Stop
------------Final Clustering Result:-------------
cluster1: (1,1) (1,2) (2,1)
cluster2: (2,3) (3,3)
cluster3: (4,5) (5,4) (6,5)
Program ended with exit code: 0
```