```
// struct testLine {
//    int movement;
//    int gender;
//    int index;
//    double vAccs[NB_VACCS];
// };

// struct model
// {
//    int numMotion;
//    double averages[NB_VACCS];
//    double stds[NB_VACCS];
//    double globalAvg;
// };

// #define NB_VACCS 600
// #define NB_MOVEMENTS 6
// #define NB_LINES_FISET 30

o──────o
│ main │
o──────o
─── *
 realClasses[NB_LINES_FISET]
 estimatedClases[NB_LINES_FISET]
 distancesAvgs[NB_MOVEMENTS]
 distancesStds[NB_MOVEMENTS]

 open fiModel in reading
 open testSet in reading

 ─── if (fiModel == NULL or testSet == NULL)
  output : " error to open files"
 ─── else
  o────────────o ↓ model[], NB_MOVEMENTS
  │ fillModel │
  o────────────o ↓ model[]
  iClass = 0
  read 1 line in testSet via file (= headers)

  ═══ do while (Not EOF AND iClass < NB_LINES_FISET)
   o──────────────o
   │ fillTestLine │
   o──────────────o ↓ testLine

   o────────────o ↓ distancesAvgs[], NB_MOVEMENTS
   │ initArray │
   o────────────o ↓ distancesAvgs[]

   o────────────o ↓ distancesStds[], NB_MOVEMENTS
   │ initArray │
   o────────────o ↓ distancesStds[]

   sumGlobalAvg = 0
   totVaccs = 0
   iMovement = 0
   iVacc = 0

   ═══ do while (iMovement < NB_MOVEMENTS)
    ═══ do while (iVAcc < NB_VACCS)

     sumGlobalAvg += testLine.vAccs[iVacc]
     distancesAvgs[iMovement] += (testLine.vAccs[iVacc] -
                 model[iMouvement].averages[iVacc])²
     lineStd = rac((testLine.vAccs[iVacc] - model[iMovement].averages[iVacc])²)
     distancesStds[iMovement] += (lineStd - model[iMovement].stds[iVacc])²
```


iVacc ++
totVaccs++
iMovement ++
iDistance = 0
do while (iDistance < NB_MOVEMENTS)
distancesAvgs[iDistance] = rac(distancesAvgs[iDistance])
distancesStds[iDistance] = rac(distancesStds[iDistance])
iDistance++
compareDistances
↓ distancesAvgs[], model[]
↓ indicator1
compareDistances
↓ distancesStd[], model[]
↓ indicator2
compareGlobalAvg
↓ sumGlobalAvg, totVaccs, , model[]
↓ indicator3
estimatedClass
↓ indicator1, indicator2, indicator3
↓ estimatedMvmt
realClasses[iClass] = testLine.movement
estimatedClasses[iClass] = estimatedMvmt
iClass++
close fiModel
close testSet
createConfusionMatrix
↓ realClasses[],estimatedClasses[], matrix[][]
↓ matrix[][]
displayResultsByClass
↓ realClasses[],estimatedClasses[], matrix[][]
displayAccuracy
↓ matrix[][]
displayConfusionMatrix
↓ matrix[][]
fillModel
↓ model[], NB_MOVEMENTS
↓ model[]
fillTestLine
↓ testLine
*
read 1 line in testSet via testLine


initArray
↓ array[], length
↓ array[]
i = 0
do while (i < length)
array[i] = 0
i++
indMinValue
↓ array[]
↓ iMin
iMin = 0
i = 1
do while (i < NB_MOVEMENTS)
if (array[i] < array[iMin])
iMin = i
i++
compareDistances
↓ distances[], model[]
↓ indicator
indMinValue
↓ distances
↓ iMin
indicator = model[iMin].numMotion
compareGlobalAvg
↓ sumGlobalAvg, totVaccs, model[]
↓ indicator3
globalAverage = sumGlobalAvg / totVaccs
distances[6]
i = 0
do while (i < NB_MOVEMENTS)
distances[i] = rac((globalAverage - model[i].globalAvg)²)
i++
indMinValue
↓ deviations
↓ iMin
indicator3 = model[iMin].numMotion
estimatedClass
↓ indicator1, indicator2, indicator3
↓ estimatedMvmt
if (indicator1 == indicator2)
estimatedMvmt = indicator1
if (indicator2 == indicator3)
estimatedMvmt = indicator2
if (indicator3 == indicator1)
estimatedMvmt = indicator3
else
estimatedMvmt = indicator1 // if the 3 indicators are different