Rapport

Contexte

Dans le cours de numérique, on a demandé de construire 3 phrases au total. Voici la tentative d'amélioration de la phrase 3.

Remarque :

Tous les résultats qui vont être présenter ont été calculer sur Linux avec gcc.

Solution de base

La solution construite au cours a été paramètre avec les distances euclidiennes avec la colonne accélération des x, accélération des y et accélération des z plus 600 colonnes dans le trainset et 60 colonnes dans le testset avec les valeurs extrêmes retirées.

Cette solution a obtenu :

23	1	44	0	0	1	23	69	33,33%
5	42	1	0	0	0	42	48	87,50%
13	0	40	3	0	15	40	71	56,34%
0	0	0	4	44	0	4	48	8,33%
0	0	0	12	36	0	36	48	75,00%
47	2	22	0	0	1	1	72	1,39%
						146	356	41,0112%

Une moyenne de résultats de 41%. On va essayer d'améliorer ça dans ce rapport.

Approche

Pour cette tentative, j'ai eu 7 grands axes d'améliorations listées ci-dessous :

- **Les colonnes prises dans le trainset** : prendre plus les colonnes.
- **Le couplement des colonnes prises** : découpler les colonnes accélérations.
- **Le nombre de colonnes du testset** : augmenter le nombre de colonnes.
- **Les critères de valeurs de extrêmes** : aucune valeur extrême n'est extrême « aucun-extrême », les valeurs sont centrées réduites avant d'être vérifiées « réduit-extrême », les valeurs sont « bornées » par des valeurs fixes « bornée-extrêmes ».
- **La méthode de filtrage de donnée** : les valeurs extrêmes sont remplacées par la moyenne « vers-moyenne », s'il y a au moins une valeur extrême sur la ligne alors cette ligne est complément ignorée « un-défaut ».
- **La méthode d'évaluation** : la distance entre 2 droites « distance euclidienne », comparer l'amplitude et la fréquence du modèle « cyclique », comparer l'histogramme du modèle « histogramme ».
- **Le calcul des moyennes et des écart-types pour chaque colonne** : pour la méthode filtrage de donnée « vers-moyenne » où on remplace les valeurs extrêmes par la moyenne. C'est la moyennes et écart-types devrait être vérifiées mais si on rajoutait d'autre colonnes il faudrait calculer nous même les moyennes et écart-types de ces nouvelles colonnes.

Développement des axes

Pour ces modifications, je vais partir de la solution de base :

Modification de la solution de base « combinaison de colonne »:
[découpler les colonnes accélérations].

En sachant qu'il y a 12 colonnes, l'ordre ne compte pas, qu'il n'y pas de remise et qu'on peut prendre 1 comme 12 colonnes Il y a $\sum_{i=1}^{12} C_{12}^{i} = 4096$ possibilités. En fait 4096, parce qu'on peut visualiser les combinaisons comme l'incrémentation d'un chiffre binaire à 12 bits où chaque bit représenterais la présence de la colonne ou non et l'exploration de chaque combination comme l'incrémentation.

Modification de la solution de base « rocher-caillou » :
[aucune valeur extrême n'est extrême], **[les valeurs sont centrées réduites avant d'être vérifiées]**, **[les valeurs sont « bornées » par des valeurs fixes]**.

J'ai trouver 3 manières qu'on pourrait considirer une valeur comme extremes. Ca fait 3 possibilités.

Modification de la solution de base « scan » :
[augmenter le nombre de colonnes] dans le testset.

En fessant varier la proportion de colonnes de testset en fonction du trainset, on peut voir le nombre de colonnes optimales. Je vais prendre 1, 0.1 et 0.5 pour ces proportions.

Modification de la solution de base « scan » :
[augmenter le nombre de colonnes] dans le trainset

On va aussi regarder quel serait le nombre de colonne optimal pour le trainset, on va prendre x1, x2, x3 plus de colonnes.

Les autre idées ne seront pas prises parce que je manque de temps.

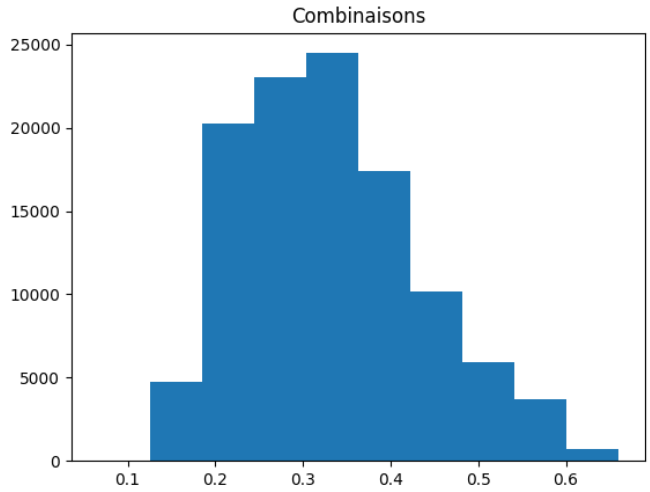
Methode d'optimations de la performance

Avec tous ces possibilités, on a 4096 * 3 * 3 * 3 = 110 592 combinaisons possibles. Avec une combinaison qui prend en moyenne 3 secondes, ca prendra 3 jours 20 heures.

Evidement tous les résultats ont été obtenu sur un serveur Linux compilé avec gcc.

Sur 110 592 combinaisons voici ce qu'on obtient :

Voici un histogramme qui represente le nombre de combinaisons en fonction de la moyenne des résultats sur tous les mouvements :



On peut observer la plupart des combinaisons sont entre 10% et 40% de précision et que le minimum est de 10% et le maximum est de 66% de précision

Il y a 6 meilleurs combinaisons :

Combinaisons de colonnes	**Nombre de colonnes du trainset**	**Nombre de colonnes du testset**	**Fonction d'extrême**	**Résultat**
011101010000	1800	180	aucun-extrême	66,05%
011101100000	1800	180	bornée-extrêmes	66,05%
110101010000	1800	180	aucun-extrême	66,05%
111000010000	1800	180	aucun-extrême	66,05%
111101010000	1800	180	aucun-extrême	66,05%
111101100000	1800	180	bornée-extrêmes	66,05%

Les 12 colonnes ont été encodées sous formes d'un chiffre binaires :

Valeur de la colonne	**Nom de la colonne**
0	ATTITUDE ROLL
1	ATTITUDE PITCH
2	ATTITUDE YAW
4	GRAVITY X
8	GRAVITY Y
16	GRAVITY Z
32	ROTATION X
64	ROTATION Y
128	ROTATION Z
256	ACCELERATION X
512	ACCELERATION Y
1024	ACCELERATION Z

Si on regarde le tableau des meilleures combinaisons, on pourrait croire que 1800 colonnes pour le trainset et 180 colonnes pour testset est meilleurs que les autres variations.

Mais si on regarde le minimum, le maximum, le maximum des résultats obtenu en fonctions du nombre de colonnes du trainset et du nombre des colonnes du testset, on obtient :

Colonnes trainset	Colonnes testset	Min	Max	Moyenne
600	60	15%	62%	33%
	300	15%	62%	36%
	600	16%	60%	36%
1200	120	13%	63%	34%
	600	10%	60%	33%
	1200	12%	56%	31%
1800	180	9%	66%	34%
	900	6%	60%	32%
	1800	12%	52%	26%

Ce qui nous intéresse, c'est la colonne des maximums. On peut voir que généralement pour chaque colonne du trainset et que les colonnes du testset augmente, au plus le maximum diminue mais que le minimum augmente. Peut-être que comparer 1800 colonnes du trainset avec 3 colonnes du testset sauterais les résultats au plafond ?

Si on observer le minimum, le maximum et la moyenne des résultats obtenur en fonctions des fonctions d'extrêmes :

Fonction extrêmes	Min	Max	Moyenne
bornée-extrêmes	6%	66%	37, 4%
réduit-extrême	12%	56%	24%
aucun-extrême	13%	66%	37, 5%

La fonction « réduit-extrême » se de marque parce que sont minimum, maximum et moyenne sont toutes inférieurs à l'autre techniques. Sinon, les 2 autres sont presque équivalentes. Il y a juste le minimum de « bornée-extrêmes » qui est inférieur à « aucun-extrême ». Donc cette dernière est la meilleure à choisir.

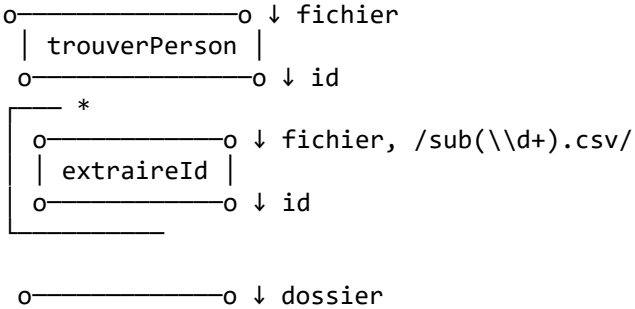
Conclusion

Donc, il y a 2 manières d'améliorer les résultats de 41% à 66% en :

- Augmentant les colonnes du trainset de 600 à 1800,
- Réduire les colonnes du testset de 600 à 180
- De filtrer les données soient « aucun-extrême » ou « bornée-extrêmes »,

Diagrammes d'actions

Phase 1

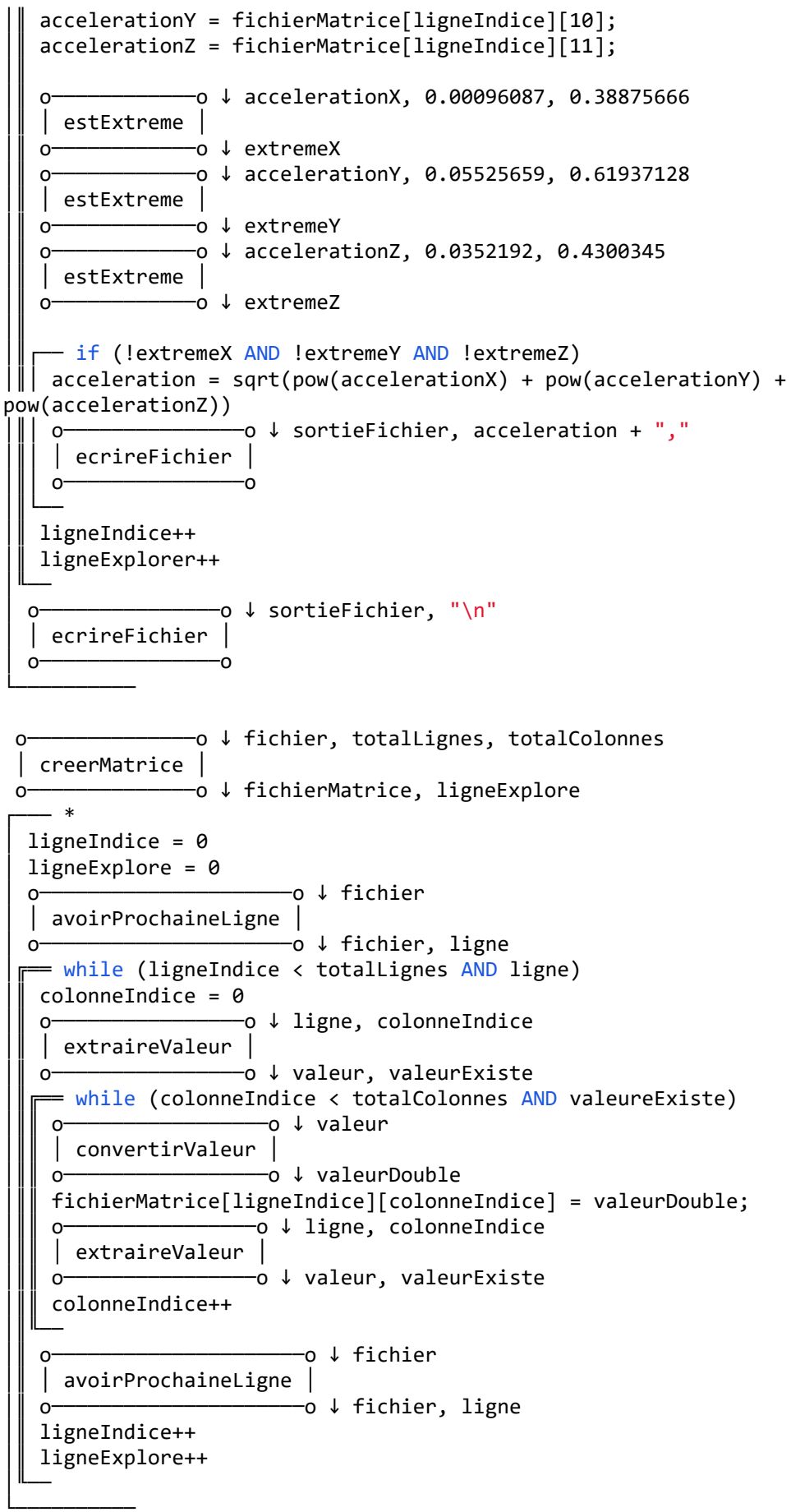


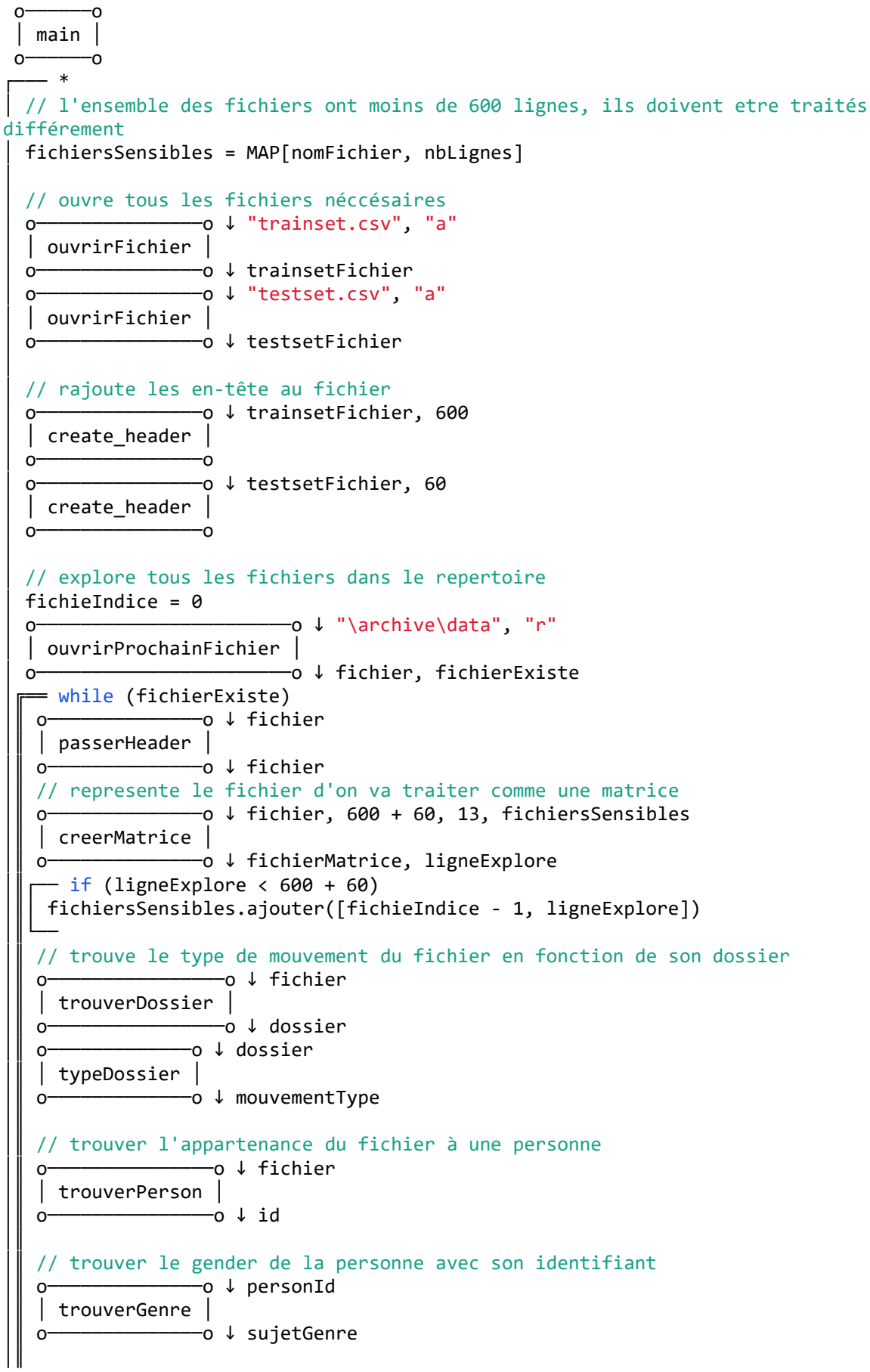
```
 | typeDossier |
 o-------------o ↓ dossierIndice
---- *
 mouvementType = ARRAY[["dws_(//d+)", DOWNSTAIR], ["jog_(//d+)", JOGGING],
["ups_(\\d+)", UPSTAIRS], ["sit_(\\d+)", SIT_DOWN], ["std_(\\d+)", STAND_UP],
["wlk_(\\d+)", WALKING]]
 dossierIndice = 0
 == while (dossierIndice < 6 AND mouvementType[dossierIndice] == dossier)
 | dossierIndice++
 ==


 o---------------o ↓ personId
 | trouverGenre |
 o---------------o ↓ sujetGenre
---- *
 o---------------o ↓ "\archive\data_subjects_info.csv", "r"
 | ouvrirFichier |
 o---------------o ↓ sujetFichier
 o---------------o ↓ sujetFichier
 | passerHeader |
 o---------------o ↓ sujetFichier
 sujetCode = 0
 sujetGenre = 0
 o---------------------o ↓ sujetFichier
 | avoirProchaineLigne |
 o---------------------o ↓ ligne
 == while (sujetIndice < personId)
 | o---------------o ↓ sujetFichier
 | | extraireSujet |
 | o---------------o ↓ sujet
 | sujetCode = sujet.code
 | sujetGenre = sujet.genre
 | o---------------------o ↓ sujetFichier
 | | avoirProchaineLigne |
 | o---------------------o ↓ ligne
 ==
 o---------------o ↓ "\archive\data_subjects_info.csv"
 | fermerFichier |
 o---------------o


 o-------------o ↓ value, average, std
 | estExtreme |
 o-------------o ↓ extreme
---- *
 extreme = value > average + (3 * std) || value < average - (3 * std)


 o-----------o ↓ fichierMatrice, ligneCommence, ligneFin, totalColonne, totalLigne,
sortieFichier
 | creerSet |
 o-----------o ↓ ligneExplorer
---- *
 ligneIndice = ligneCommence
 ligneExplorer = 0
 == while (ligneIndice < (ligneCommence + ligneFin) AND ligneExplorer <
totalLigne AND ligneIndice < totalLigne)
 || accelerationX = fichierMatrice[ligneIndice][9];
```


accelerationY = fichierMatrice[ligneIndice][10];
accelerationZ = fichierMatrice[ligneIndice][11];
estExtreme
accelerationX, 0.00096087, 0.38875666
extremeX
estExtreme
accelerationY, 0.05525659, 0.61937128
extremeY
estExtreme
accelerationZ, 0.0352192, 0.4300345
extremeZ
if (!extremeX AND !extremeY AND !extremeZ)
acceleration = sqrt(pow(accelerationX) + pow(accelerationY) +
pow(accelerationZ))
sortieFichier, acceleration + ","
ecrireFichier
ligneIndice++
ligneExplorer++
sortieFichier, "\n"
ecrireFichier
fichier, totalLignes, totalColonnes
creerMatrice
fichierMatrice, ligneExplore
*
ligneIndice = 0
ligneExplore = 0
fichier
avoirProchaineLigne
fichier, ligne
while (ligneIndice < totalLignes AND ligne)
colonneIndice = 0
ligne, colonneIndice
extraireValeur
valeur, valeurExiste
while (colonneIndice < totalColonnes AND valeureExiste)
valeur
convertirValeur
valeurDouble
fichierMatrice[ligneIndice][colonneIndice] = valeurDouble;
ligne, colonneIndice
extraireValeur
valeur, valeurExiste
colonneIndice++
fichier
avoirProchaineLigne
fichier, ligne
ligneIndice++
ligneExplore++

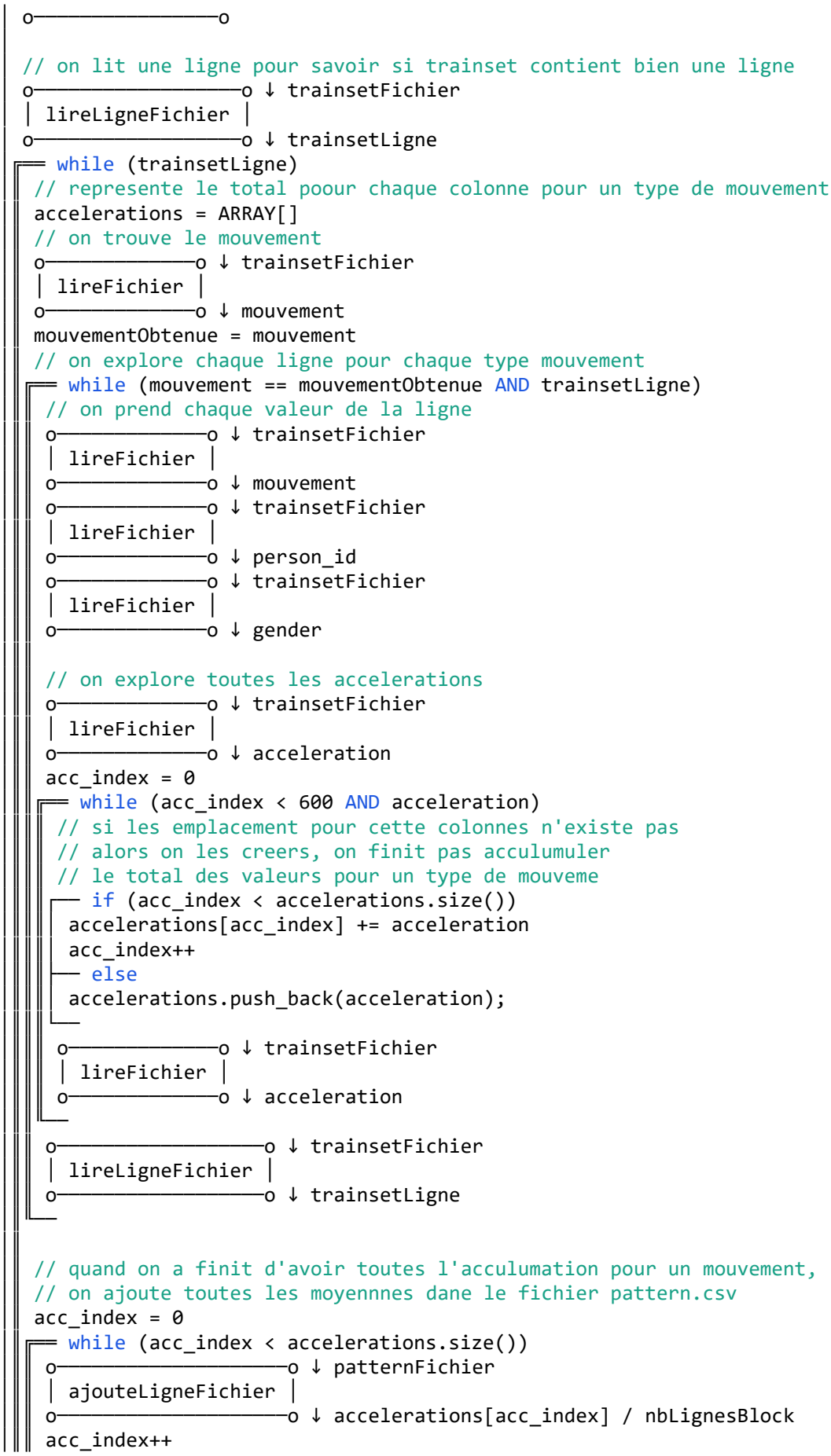

main
// l'ensemble des fichiers ont moins de 600 lignes, ils doivent etre traités différement
fichiersSensibles = MAP[nomFichier, nbLignes]
// ouvre tous les fichiers néccésaires
ouvrirFichier
"trainset.csv", "a"
trainsetFichier
"testset.csv", "a"
ouvrirFichier
testsetFichier
// rajoute les en-tête au fichier
create_header
trainsetFichier, 600
create_header
testsetFichier, 60
// explore tous les fichiers dans le repertoire
fichieIndice = 0
ouvrirProchainFichier
"\archive\data", "r"
fichier, fichierExiste
while (fichierExiste)
passerHeader
fichier
fichier
// represente le fichier d'on va traiter comme une matrice
creerMatrice
fichier, 600 + 60, 13, fichiersSensibles
fichierMatrice, ligneExplore
if (ligneExplore < 600 + 60)
fichiersSensibles.ajouter([fichieIndice - 1, ligneExplore])
// trouve le type de mouvement du fichier en fonction de son dossier
trouverDossier
fichier
typeDossier
dossier
mouvementType
// trouver l'appartenance du fichier à une personne
trouverPerson
fichier
id
// trouver le gender de la personne avec son identifiant
trouverGenre
personId
sujetGenre

```
// ecrie les 3 premieres colonne et creer le trainset
o————————————o ↓ trainsetFichier, mouvementType + "," + sujetGenre + "," +
fichieIndice
| ecrireFichier |
o————————————o
o————————————o ↓ fichierMatrice, 0, 600, 13, 600, trainsetFichier
| creerSet |
o————————————o ↓ trainsetligneExplorer
// si le trainset à assez de ligne pour creer un testset
┌── if (trainsetligneExplorer == 600)
| // ecrie les 3 premieres colonne et creer le testset
| o————————————o ↓ testsetFichier, mouvementType + "," + sujetGenre + "," +
fichieIndice
| | ecrireFichier |
| o————————————o
| o————————————o ↓ fichierMatrice, 600, 660, 13, 660, testsetFichier
| | creerSet |
| o————————————o ↓ testsetligneExplorer
└──
o————————————————————o ↓ "\archive\data", "r"
| ouvrirProchainFichier |
o————————————————————o ↓ fichier, fichierExiste
fichieIndice++
└──

// ferme tous les fichiers utilisés
o————————————o ↓ "trainset.csv"
| fermerFichier |
o————————————o
o————————————o ↓ "testset.csv"
| fermerFichier |
o————————————o
```

Phase 2

```
o————————o
| pattern |
o————————o
┌── *
| // ouvre tous les fichier nécessaires
| o————————————o ↓ "pattern.csv", "w"
| | ouvrirFichier |
| o————————————o ↓ patternFichier
| o————————————o ↓ "trainset.csv", "r"
| | ouvrirFichier |
| o————————————o ↓ trainsetFichier
|
| // creer l'en-tête pour pattern.csv
| index = 0;
| o————————————o ↓ patternFichier, "Mouvement,"
| | ajouterFichier |
| o————————————o
| ╔══ while (index < 600)
| | o————————————o ↓ patternFichier, "Vacc,"
| | | ajouterFichier |
| | o————————————o
| | index++
| ╚══
| o————————————o ↓ patternFichier, "\n"
| | ajouterFichier |
```


o——————o
// on lit une ligne pour savoir si trainset contient bien une ligne
o——————o ↓ trainsetFichier
lireLigneFichier
o——————o ↓ trainsetLigne
while (trainsetLigne)
// represente le total poour chaque colonne pour un type de mouvement
accelerations = ARRAY[]
// on trouve le mouvement
o——————o ↓ trainsetFichier
lireFichier
o——————o ↓ mouvement
mouvementObtenue = mouvement
// on explore chaque ligne pour chaque type mouvement
while (mouvement == mouvementObtenue AND trainsetLigne)
// on prend chaque valeur de la ligne
o——————o ↓ trainsetFichier
lireFichier
o——————o ↓ mouvement
o——————o ↓ trainsetFichier
lireFichier
o——————o ↓ person_id
o——————o ↓ trainsetFichier
lireFichier
o——————o ↓ gender
// on explore toutes les accelerations
o——————o ↓ trainsetFichier
lireFichier
o——————o ↓ acceleration
acc_index = 0
while (acc_index < 600 AND acceleration)
// si les emplacement pour cette colonnes n'existe pas
// alors on les creers, on finit pas acculumuler
// le total des valeurs pour un type de mouveme
if (acc_index < accelerations.size())
accelerations[acc_index] += acceleration
acc_index++
else
accelerations.push_back(acceleration);
o——————o ↓ trainsetFichier
lireFichier
o——————o ↓ acceleration
o——————o ↓ trainsetFichier
lireLigneFichier
o——————o ↓ trainsetLigne
// quand on a finit d'avoir toutes l'acculumation pour un mouvement,
// on ajoute toutes les moyennnes dane le fichier pattern.csv
acc_index = 0
while (acc_index < accelerations.size())
o——————o ↓ patternFichier
ajouteLigneFichier
o——————o ↓ accelerations[acc_index] / nbLignesBlock
acc_index++

```
o──────────────────o ↓ patternFichier
 ajouteLigneFichier
o──────────────────o ↓ '\n'
```

```
o────────────────o ↓ patternFichier
 fermerFichier
o────────────────o
o────────────────o ↓ trainsetFichier
 fermerFichier
o────────────────o
```

Phase 3

```
// explore pattern.csv et trouve la ligne qui correspond à mouvementType
 o──────────────────o ↓ mouvementType
  trouveAccelerations
 o──────────────────o ↓ accelerations, nbAcceleration
──── *
 // l'ensemble des accelerations qui correspond à mouvementType
 accelerations = ARRAY[DOUBLE]

 // ouvre le fichier "pattern.csv" en lecture
 o────────────────o ↓ "pattern.csv", "r"
  ouvrirFichier
 o────────────────o ↓ patternFichier

 // passe l'en-tête, il n'est pas nécesaire
 o──────────────o ↓ patternFichier
  passerEnTete
 o──────────────o ↓ patternFichier

 // recherche et s'arrette à la ligne correspodant à mouvementType
 o──────────────────o ↓ patternFichier
  passerProchaineLigne
 o──────────────────o ↓ patternLigne, patterLigne, ligneExiste
 o────────────────o ↓ patternLigne
  extraireValeur
 o────────────────o ↓ patternLigne, mouvemenTypeLigne, valeurExiste
╔══ while (valeurExiste AND ligneExiste AND mouvementType < mouvemenTypeLigne)
 o──────────────────o ↓ patternFichier
  passerProchaineLigne
 o──────────────────o ↓ patterLigne, ligneExiste
 o────────────────o ↓ patternLigne
  extraireValeur
 o────────────────o ↓ patternLigne, mouvemenTypeLigne, valeurExiste
╚══

 // ici mouvementType vaut mouvemenTypeLigne (mouvementType == mouvemenTypeLigne)
 // Ca veut dire que patternLigne vaut la ligne dont on extraire toutes
 // les accelerations qui correspond à mouvementType
 nbAcceleration = 0
 o────────────────o ↓ patternLigne
  extrainValeur
 o────────────────o ↓ acceleration, valeurExiste
╔══ while (valeurExiste)
 accelerations[nbAcceleration] = acceleration
 nbAcceleration++
 o────────────────o ↓ patternLigne
  extrainValeur
```

```
o──────────────o ↓ acceleration, valeurExiste

// ferme le fichier utilisé
o──────────────o ↓ patternFichier
| fermerFichier |
o──────────────o

// trouve le minimum entre 2 valeurs
o─────o ↓ a, b
| min |
o─────o ↓ c
── *
── if (a < b)
c = a
── else
c = b

// tous commence ici
o─────────────o
| evaluation |
o─────────────o
── *
// ouvre tous les fichiers nécesaires
o──────────────o ↓ "testset.csv", "r"
| ouvrirFichier |
o──────────────o ↓ testsetFichier

// passer l'en-tête du testset
o──────────────o ↓ testsetFichier
| passerEnTete |
o──────────────o ↓ testsetFichier

// pour chaque ligne dans testset
o──────────────────────o ↓ testsetFichier
| passerProchainLigne |
o──────────────────────o ↓ testsetLigne, ligneExiste
== while (ligneExiste)
// extrait le type de mouvement, le genre et index de chaque ligne
o──────────────o ↓ testsetLigne
| extraireValeur |
o──────────────o ↓ testsetLigne, mouvement, valeurExiste
o──────────────o ↓ testsetLigne
| extraireValeur |
o──────────────o ↓ testsetLigne, genre, valeurExiste
o──────────────o ↓ testsetLigne
| extraireValeur |
o──────────────o ↓ testsetLigne, index, valeurExiste

distanceMin = MX;
movementMin = MX;

// compare la ligne du testset à chaque ligne du pattern
mouvemenTypeIndice = 0
== while (mouvemenTypeIndice < 6)
// trouve la ligne qui correspond à mouvementType
```

```
o──────────────────────o ↓ mouvemenTypeIndice
│ trouveAccelerations  │
o──────────────────────o ↓ accelerations, nbAcceleration

total = 0

// compare la ligne de testset avec celle de pattern
iAcceleration = 0
╔══ while (iAcceleration < nbAcceleration)
╠ // extrait la valeur testset de la valeur accelerations correspondante
╠ o──────────────────o ↓ accelerations
╠ │ extraireValeur   │
╠ o──────────────────o ↓ accelerations, acceleration, valeurExiste
╠ o──────────────────o ↓ testsetLigne
╠ │ extraireValeur   │
╠ o──────────────────o ↓ testsetLigne, valeur, valeurExiste
╠
╠ total += pow(acceleration - valeur, 2)
╠ mouvemenTypeIndice++
╚══

// trouve la distance euclidienne minimal pour cette ligne du testset
╔══ if (sqrt(total) < distanceMin)
╠ distanceMin = sqrt(total)
╠ movementMin = mouvemenTypeIndice
╚══

o──────────────────────o ↓ testsetFichier
│ passerProchainLigne  │
o──────────────────────o ↓ testsetLigne, ligneExiste
╚══

sortir "le mouvement: ", mouvement, "a été devinier comme", movementMin

// ferme tous les fichiers ouvert
o──────────────────o ↓ testsetFichier
│ fermerFichier    │
o──────────────────o
```