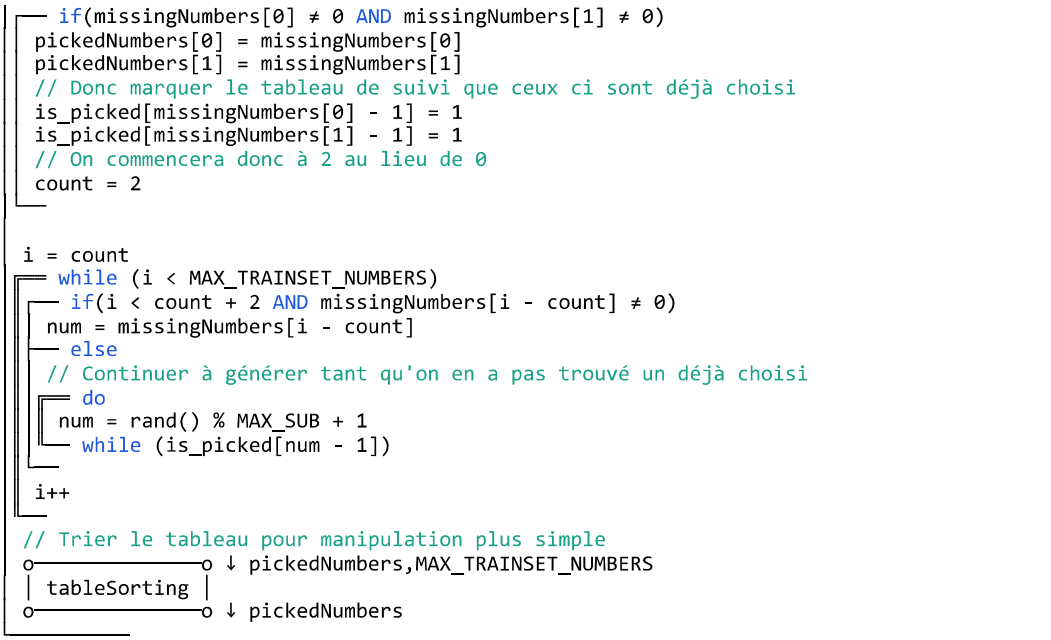
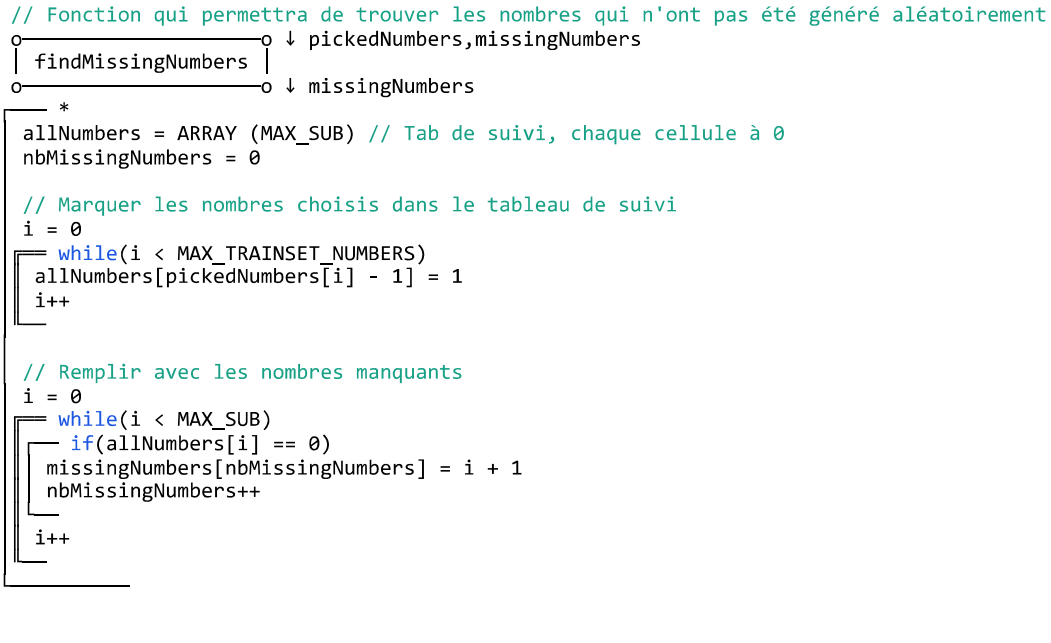
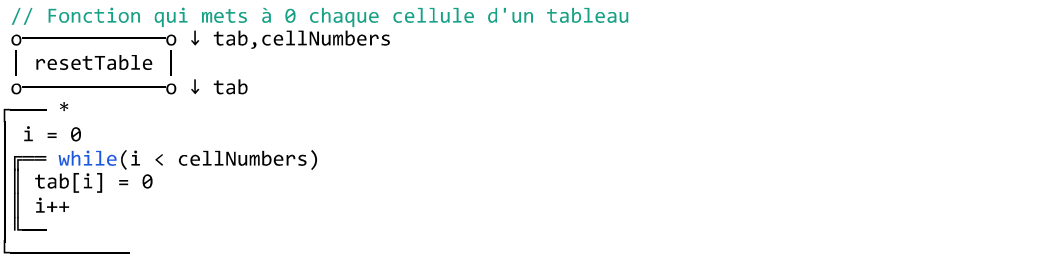
```
// Définir des constantes

MAX_NUMBERS 1803 // car 601 lignes * 3 = 1803
MAX_LINES 600
MAX_PARAM_ACC_CALCUL 3 // Pour les variables x, y et z
MAX_FILES 15
MAX_SUB 24
MAX_TRAINSET_NUMBERS 22 // Car on prendra 90% des 24 subs au hasard donc 22
MAX_TESTSET_NUMBERS 2 // idem mais pour 10% des 24
MAX_CHAR_NAME_MOVEMENTS 50
FILE_PATH "data/A_DeviceMotion_data/%s/sub_%d.csv"
TRAINSET_NAME "trainSet.csv"
TESTSET_NAME "testSet.csv"

// Définir une structure (fiche) subStats qui correspondra au Stats d'un Sub
// Cette fiche contiendra un genre, un index et un tableau avec tous ses Vacc

// -----------
// Fonction principale du programme
o------o
| Main |
o------o
------ *
// Initialisation des variables pour manipulation
fileNames = ARRAY(MAX_FILES)
fileNames[0] = "dws_1"
fileNames[1] = "dws_2"
...
fileNames[14] = "wlk_15"

genderNumber = ARRAY(MAX_SUB) // Tableau pour les genres des Sub
testSetNumbers = allocation pour un tableau 2 dimmension de MAX_FILES * MAX_TESTSET_NUMBERS
trainSetNumbers = allocation pour un tableau 2 dimmension de MAX_FILES * MAX_TRAINSET_NUMBERS
is_picked = ARRAY(MAX_SUB) // Tableau qui permettra de suivre l'aléatoire où chaque cellule est à 0

srand = initialiser la séquence d'aléatoire pour générer des nombres

trainSet = allocation mémoire pour un tableau 2 dimensions de structure subStats (fiche) de MAX_FILES
testSet = allocation mémoire pour un tableau 2 dimensions de structure subStats (fiche) de MAX_FILES
---- if(trainSet == NULL OR testSet == NULL)
Sortir "Erreur d'allocation de mémoire"
---- else
// Extraire les numéros des genre pour chaque Sub dans le tableau genderNumber
o----------------o ↓ genderNumber
| extractGender |
o----------------o ↓ genderNumber

// Boucle principale qui manipulera les 360 fichiers csv
iFile = 0
==== while(iFile < MAX_FILES) // Boucle qui en réalité est un for

// Initialiser pour chaque dossier , ses colonnes pour stocker ses Sub
trainSet[iFile] = allocation mémoire pour les colonnes du tableau pour MAX_NOMBRES_TRAINSET
testSet[iFile] = allocation mémoire pour les colonnes du tableau pour MAX_NOMBRES_TESTSET
---- if(trainSet[iFile] == NULL OR testSet[iFile] == NULL)
Sortir "Erreur d'allocation de mémoire"
---- else

// Générer aléatoirement les numéros des Sub pour le TrainSet et TestSet

o----------------------o ↓ trainSetNumbers[iFile],is_picked,trainSetNumbers[iFile]
| generateRandomSubs |
o----------------------o ↓ trainSetNumbers[iFile],testSetNumbers[iFile]

o----------------------o ↓ trainSetNumbers[iFile],trainSetNumbers[iFile]
| findMissingNumbers |
o----------------------o ↓ trainSetNumbers[iFile],trainSetNumbers[iFile]

o-------------o ↓ is_picked,MAX_SUB
| resetTable |
o-------------o ↓ is_picked

// Traitement des tableau TrainSet et TestSet pour calculer leurs Vacc
iSub = 0
==== while(iSub < MAX_TRAINSET_NUMBERS) // Boucle for

o----------------o ↓ fileNames,iFile,trainSetNumbers[iFile][iSub]
| subProcessing |
o----------------o ↓ ficheSub

trainSet[iFile][iSub] = ficheSub // est une variable de type struct StatsSub
trainSet[iFile][iSub].gender = genderNumber[trainSetNumbers[iFile][iSub] - 1]
iSub++
----


iSub = 0
==== while(iSub < MAX_TESTSET_NUMBERS) // Boucle for
```

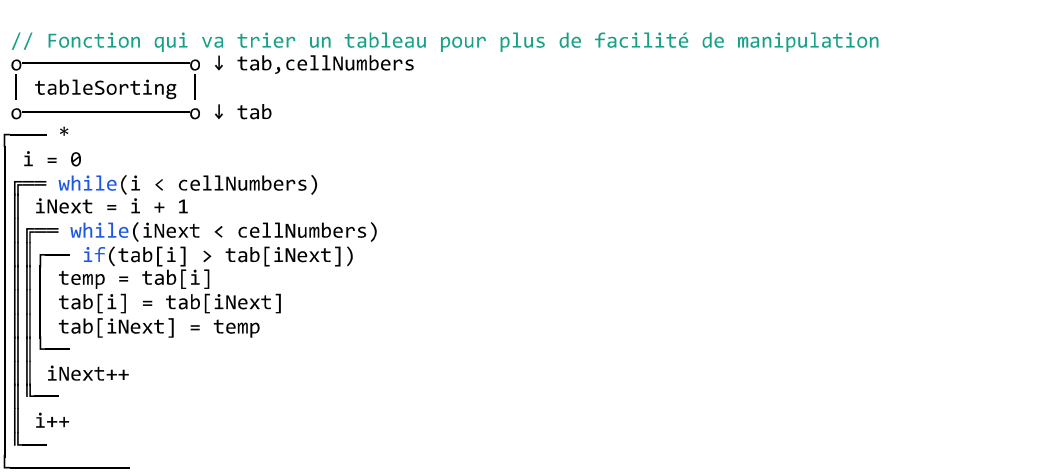
```
    o──────────────o ↓ fileNames,iFile,testSetNumbers[iFile][iSub]
    │ subProcessing │
    o──────────────o ↓ ficheSub

    testSet[iFile][iSub] = ficheSub // est une variable de type struct StatsSub
    testSet[iFile][iSub].gender = genderNumber[testSetNumbers[iFile][iSub] - 1]
    iSub++

  iFile++

  // Imprimer les résultats dans des csv
  o──────────o ↓ testSet,nameMouvement,MAX_TESTSET_NUMBERS,NAME_TESTSET
  │ printCSV │
  o──────────o
  o──────────o ↓ trainSet,nameMouvement,MAX_TRAINSET_NUMBERS,NAME_TRAINSET
  │ printCSV │
  o──────────o

  Libérer l'espace mémoire de trainSet et testSet
  Sortir "Traitement terminé"


// Fonction qui permettra d'extraire les numéros des genres de chaque Sub
o──────────────o ↓ genderNumber
│ extractGender │
o──────────────o ↓ genderNumber
─── *
file = ouverture du fichier "data_subjects_info.csv"
─── if(file == NULL)
sortir "Impossible d'ouvrir le fichier"
─── else
line = ARRAY (1024)
i = 0
fgets(...) // Extraire la première ligne qui sont les titres
═══ while(fgets(line,sizeof(line),file))
sscanf_s(line,"%*f,%*f,%*f,%*f,%lf",&last_numbers[i])
i++

Fermer le dossier


// Fonction qui va traiter un fichier csv avec des indices recu en params
o──────────────o ↓ nameMouvement,iFile,iSub
│ subProcessing │
o──────────────o ↓ ficheSub
─── *
numbers = ARRAY(MAX_NUMBERS)

o──────────────────────────o ↓ numbers,nameMouvement,iFile,iSub
│ extractLastThreeNumbers │
o──────────────────────────o ↓ numbers

Créer une struct (fiche) de type subStats nommé ficheSub
ficheSub.index = iSub

o────────────────────o ↓ numbers,ficheStatsSub.stats
│ obtainingVectors │
o────────────────────o ↓ ficheSub.stats

Libérer l'espace mémoire pour numbers


// Fonction qui va extraire un fichier avec son indice de mouvement et de sub dans un tableau tous ses nombre nécessaire
o──────────────────────────o ↓ numbers,nameMouvement,iFile,iSub
│ extractLastThreeNumbers │
o──────────────────────────o ↓ numbers
─── *
fileName = ARRAY(MAX_CHAR_NAME_MOVEMENTS)
fileName = assosier le iFile et iSub à PATH_FILE pour avoir le sub correspondant

file = ouvrir fileName
─── if(file == NULL)
Sortir "Impossible d'ouvrir le fichier"
─── else
line = ARRAY(1024) // Pour extraire les lignes du csv
i = 0
lineNumber = 0
// Ouvrir chaque ligne du csv et ne prendre que les 3 derniers nombres
═══ while(fgets(line,sizeof(line),file) AND lineNumber < MAX_NUMBERS_ACC_CALCUL)
sscanf_s(line, "%*f,%*f,....,%lf,%lf,%lf", &numbers[i],&numbers[i+1],&numbers[i+2])
i += 3
lineNumber ++
```

```
Fermer le fichier

// Fonction qui va renvoyer dans le tabl stats du user son Vacc pour chaque ligne
o-------------------o ↓ numbers,ficheStatsSub.stats
| obtainingVectors |
o-------------------o ↓ ficheSub.stats
*
iVariable = 1
calcul = ARRAY(MAX_PARAM_ACC_CALCUL)
iResult = 0
iNumber = 0
while(iNumber < MAX_NUMBERS) // Boucle for
 if(iVariable ≤ MAX_PARAM_ACC_CALCUL)
  calcul[iVariable-1] = numbers[iNumber]
  if(iVariable == MAX_PARAM_ACC_CALCUL)
   o-----------------------o ↓ calcul
   | calculAccelerationVector |
   o-----------------------o ↓ calcul,result
   ficheStatsSub.stats[iResult] = result
   iResult++
   iVariable = 0
 iVariable++
 iNumber++

// Fonction pour calculer un vecteur Acc
o-----------------------o ↓ calcul
| calculAccelerationVector |
o-----------------------o ↓ calcul,resultat
*
sum = 0
iVariable = 0
while(iVariable < MAX_PARAM_ACC_CALCUL)
 sum += calcul[iVariable] * calcul[iVariable]
 iVariable++
resultat = [sum]racine


// Fonction qui va imprimer dans un csv un tableau de type StatsSub
o-----------o ↓ data,fileNames,cellNumbers,fileName
| printCSV |
o-----------o
*
file = ouvrir fileName.csv
 if(file == NULL)
 sortir "Erreur lors de l'ouverture du fichier"
 else

 file = Ecrire l'en-téte du fichier du csv → "Mouvement,Gender,Index"
 iLine = 0
 while(iLine < MAX_Lines)
  file += ",VaccAcc " + iLine
  iLine++
 index = 0
 iFile = 0
 while(iFile < MAX_FILES)
  while(iSub < cellNumbers)
   file += nameMouvement[iFile] + "," + data[iFile][iSub].gender + "," + index
   iLine = 0
   while(iLine < MAX_NUMBERS_ACC_CALCUL AND data[iFile][iSub].stats[iLine] ≠ 0)
    file += "," + data[iFile][iSub].stats[iLine]
    iLine++
   file += "\n"
   iSub++
   index++
  iFile++
 Fermeture du fichier

// Fonction pour générer des numéros de Sub aléatoire
o--------------------o ↓ pickedNumbers,is_picked,missingNumbers
| generateRandomSubs |
o--------------------o ↓ pickedNumbers,is_picked,missingNumbers
*
count = 0
// Vérifier si il y a des nombres manquants, si oui les mettre prioritaire pour la suite
```


if(missingNumbers[0] ≠ 0 AND missingNumbers[1] ≠ 0)
pickedNumbers[0] = missingNumbers[0]
pickedNumbers[1] = missingNumbers[1]
// Donc marquer le tableau de suivi que ceux ci sont déjà choisi
is_picked[missingNumbers[0] - 1] = 1
is_picked[missingNumbers[1] - 1] = 1
// On commencera donc à 2 au lieu de 0
count = 2
i = count
while (i < MAX_TRAINSET_NUMBERS)
if(i < count + 2 AND missingNumbers[i - count] ≠ 0)
num = missingNumbers[i - count]
else
// Continuer à générer tant qu'on en a pas trouvé un déjà choisi
do
num = rand() % MAX_SUB + 1
while (is_picked[num - 1])
i++
// Trier le tableau pour manipulation plus simple
tableSorting
↓ pickedNumbers,MAX_TRAINSET_NUMBERS
↓ pickedNumbers


// Fonction qui permettra de trouver les nombres qui n'ont pas été généré aléatoirement
findMissingNumbers
↓ pickedNumbers,missingNumbers
↓ missingNumbers
*
allNumbers = ARRAY (MAX_SUB) // Tab de suivi, chaque cellule à 0
nbMissingNumbers = 0
// Marquer les nombres choisis dans le tableau de suivi
i = 0
while(i < MAX_TRAINSET_NUMBERS)
allNumbers[pickedNumbers[i] - 1] = 1
i++
// Remplir avec les nombres manquants
i = 0
while(i < MAX_SUB)
if(allNumbers[i] == 0)
missingNumbers[nbMissingNumbers] = i + 1
nbMissingNumbers++
i++


// Fonction qui mets à 0 chaque cellule d'un tableau
resetTable
↓ tab,cellNumbers
↓ tab
*
i = 0
while(i < cellNumbers)
tab[i] = 0
i++


// Fonction qui va trier un tableau pour plus de facilité de manipulation
tableSorting
↓ tab,cellNumbers
↓ tab
*
i = 0
while(i < cellNumbers)
iNext = i + 1
while(iNext < cellNumbers)
if(tab[i] > tab[iNext])
temp = tab[i]
tab[i] = tab[iNext]
tab[iNext] = temp
iNext++
i++