Structures :

Client { statut, duréeAttente, duréeTraitement}

Stations { clients } File { Clients }

o───────────────────────o ↓ γPrioritaire,γOrdinaire,a,c,m,x0

│ simulationFileAttente │

o───────────────────────o ↓ nbStationsOptimal

┌─── \* **simulationFileAttente**

│ nbStations = nbStationsMin

│╔══ do while(nbStations ≤ nbStationsMax)

│║ longueurFile = 0

│║ duréeTotaleClientOrdinaire = 0

│║ duréeTotaleClientPrioritaire = 0

│║ duréeTotaleClientAbsolu = 0

│║ fileCumulée = 0 // A voir avec FE

│║ o─────────────o

│║ │ initStation │

│║ o─────────────o ↓ stations

│║ temps = 1

│║╔══ do while (temps ≤ tempsSimulation)

│║║ // Incrémenter durée totale des clients selon leur type + Retirer de la file les clients trop impatients

│║║ o───────────────────o ↓ file, longueurFile

│║║ │ gestionImpatience │

│║║ o───────────────────o ↓ file, longueurFile

│║║ // Générer x ordinaires, y relatifs selon les params poisson

│║║ o────────────────────────o ↓ γPrioritaire,γOrdinaire,a,c,m,x0

│║║ │ genererNouveauxClients │

│║║ o────────────────────────o ↓ clients,nbArrivées,x0

│║║ longueurFile += nbArrivées

│║║ ind = 0

│║║ // Trier la file avec les nouveaux clients > (prio >>> normaux)

│║║ o─────────────o ↓ file, longueurFile, clients, nbArrivée

│║║ │ gestionFile │

│║║ o─────────────o ↓ file, longueurFile

│║║ // Mettre les clients prio en station, et gérer les clients ejectés

│║║ o──────────────────────────o ↓ stations, nbStations, file, longueurFile

│║║ │ gestionClientPrioritaire │

│║║ o──────────────────────────o ↓ stations, file

│║║╔══ do while(ind < nbStations)

│║║║┌── if(stations[ind] == 0)

│║║║│┌── if(longieurFile ≠ 0)

│║║║││ longueurFile --

│║║║││ o────────────────────────o ↓ a,c,m,x0

│║║║││ │ genererDuréeTraitement │

│║║║││ o────────────────────────o ↓ duréeTraitement,x0

│║║║││ stations[ind].duréeTraitement = duréeTraitement

│║║║││ o─────────────o ↓ file

│║║║││ │ avancerFile │

│║║║││ o─────────────o ↓ file

│║║║││ stations[ind].duréeTraitement -- // On accéde à durée trt de client contenu dans stations │║║║│└── │║║║├── else

│║║║│ stations[ind].duréeTraitement -- // idem

│║║║└──

│║║║ ind ++

│║║╙──

│║║ fileCumulée += file

│║║ temps ++

│║└──

│║ couts[nbStations - nbStationsMin] = A \* nbStations \* B \* fileCumulée / tempsSimulation // A adapter

│║ nbStations ++

│╙──

│ o──────────────────o ↓ couts, nbStationsMin, nbStationsMax

│ │ rechercheCoutMin │

│ o──────────────────o ↓ nbStationsOptimal

└──────────

// gérer les incr sur les durées totale ⇒ Parcourir la file après la gestion impatience

// incr de 1 dans bonne catégorie en parcourant file

o───────────────────o ↓ file, longueurFile

│ gestionImpatience │

o───────────────────o ↓ file, longueurFile

┌─── \* **gestionImpatience**

│ iFile = 0

│╔══ do while (iFile < longueurFile)

│║ file[iFile].duréeAttente ++

│║

│║┌── if(file[iFile].statut == "ordinaire")

│║│ duréeTotaleClientOrdinaire ++

│║├── else if(file[iFile].statut == "prio\_relatif")

│║│ duréeTotaleClientPrioritaire ++

│║├── else

│║│ duréeTotaleClientAbsolu ++

│║└──

│║┌── if(file[iFile].duréeAttente ≥ 10 AND iFile ≥ 3)

│║│ o─────────────────o ↓ iFile, file, longueurFile

│║│ │ supprimerClient │

│║│ o─────────────────o ↓ file, longueurFile

│║└──

│║ iFile ++

│╙──

└──────────

// Créer une liste de clients qui contient les clients ordinaires et prioritaires générés o────────────────────────o ↓ γPrioritaire,γOrdinaire,a,c,m,x0

│ genererNouveauxClients │

o────────────────────────o ↓ clients,nbArrivées,x0

┌─── \* **generationTypeClient**

│ clients = []

│ o───────────────────o ↓ γ,a,c,m,x0

│ │ générerNbArrivées │

│ o───────────────────o ↓ nbArrivéesClientsOrdinaires,x0

│ o───────────────────o ↓ γ,a,c,m,x0

│ │ générerNbArrivées │

│ o───────────────────o ↓ nbArrivéesClientsPrioritaires,x0

│ nbAbsolus = nbArrivéesClientsPrioritaires \* 0.3

│ nbRelatifs = nbArrivéesClientsPrioritaires - nbAbsolus

│ i = 0

│╔══ do while (i < nbAbsolus )

│║ client.statut = "absolu"

│║ client.duréeAttente = 0

│║ clients.append(client)

│║ i++

│╙──

│ i = 0

│╔══ do while (i < nbRelatifs )

│║ client.statut = "relatif"

│║ client.duréeAttente = 0

│║ clients.append(client)

│║ i++

│╙──

│ i = 0

│╔══ do while (i < nbArrivéesClientsOrdinaires)

│║ client.statut = "ordinaire"

│║ client.duréeAttente = 0

│║ clients.append(client)

│║ i++

│╙──

│ nbArrivées = nbArrivéesClientsOrdinaires + nbArrivéesClientsPrioritaires

└──────────

o───────────────────o ↓ γ,a,c,m,x0

│ générerNbArrivées │

o───────────────────o ↓ nbArrivées,x0

┌─── \* **générerNbArrivées**

│ o───────────────────────────o ↓ a,c,m,x0

│ │ générationNombreAléatoire │

│ o───────────────────────────o ↓ x1

│ u1 = x1 / m

│ x0 = x1

│ k = 0

│ fx = 0

│╔══ do

│║ o────────────o ↓ k,γ

│║ │ loiPoisson │

│║ o────────────o ↓ probabilité

│║ fx += probabilité

│║ k++

│╙── while (u1 ≥ fx)

│ nbArrivées = k – 1

└──────────

o───────────────────────────o ↓ a,c,m,x0 │ générationNombreAléatoire │ o───────────────────────────o ↓ x1

┌─── \* **générationNombreAléatoire**

│ x1 = (a \* x0 + c) % m

└──────────

o────────────o ↓ k,γ

│ loiPoisson │ o────────────o ↓ probabilité

┌─── \* **loiPoisson**

│ probabilité = ((e^-γ) \* (γ^k)) / k!

└──────────

// Ajouter et trier la file avec les nouveaux clients > (prio >>> normaux)

o─────────────o ↓ file, longueurFile, clients, nbArrivée

│ gestionFile │

o─────────────o ↓ file, longueurFile

┌─── \* **gestionFile**

│ iArrivée = 0

│╔══ do while(iArrivée < nbArrivée)

│║ o─────────────────o ↓ clients, file, longueurFile

│║ │ ajoutClientFile │

│║ o─────────────────o ↓ clients, file, longueurFile

│╙──

└──────────

o──────────────────────────o ↓ stations, nbStations, file, longueurFile

│ gestionClientPrioritaire │

o──────────────────────────o ↓ stations, file

┌─── \* **gestionClientPrioritaire**

│ iClient = 0

│ clientsEjectés = []

│ nbClientsEjectés = 0

│╔══ do while (iClient < longueurFile AND file[iClient].statut == "absolu")

│║ tempsTraitementMax = LV

│║ numStationMax = -1

│║ iStation = 0

│║╔══ do while (iStation < nbStations)

│║║┌── if (stations[iStation].statut == "ordinaire" and stations[iStation].duréeTraitement > tempsTraitementMax) │║║│ numStationMax = iStation

│║║│ tempsTraitementMax = stations[iStation].duréeTraitement

│║║└──

│║╙──

│║┌── if (numStation ≠ -1)

│║│ o────────────────────────o ↓ a,c,m,x0

│║│ │ genererDuréeTraitement │

│║│ o────────────────────────o ↓ duréeTraitement,x0

│║│ clientsEjectés.append(stations[numStation])

│║│ stations[numStation] = file[iClient]

│║│ stations[numStation].duréeTraitement = duréeTraitement

│║│ nbClientsEjectés++

│║└──

│║ iClient++

│╙──

│ iClientEjecté = 0

│╔══ do while (iClientEjecté < nbClientsEjectés)

│║ file[iClientEjecté] = clientsEjectés[iClientEjecté]

│║ iClientEjecté++

│╙──

└──────────

o────────────────────────o ↓ a,c,m,x0

│ genererDuréeTraitement │ o────────────────────────o ↓ duréeTraitement,x0

┌─── \* **genererDuréeTraitement**

│ o───────────────────────────o ↓ a,c,m,x0

│ │ générationNombreAléatoire │ │ o───────────────────────────o ↓ x1

│ u1 = x1 / m

│ x0 = x1

│ proba = 2 / 62

│┌── if (u1 < proba)

││ duréeTraitement = 6

│├── else

││ proba += 3 / 62

││┌── if (u1 < proba)

│││ duréeTraitement = 5

││├── else

│││ proba += 4 / 62

│││┌── if (u1 < proba)

││││ duréeTraitement = 4

│││├── else

││││ proba += 11 / 62

││││┌── if (u1 < proba)

│││││ duréeTraitement = 3

││││├── else

│││││ proba += 18 / 62

│││││┌── if (u1 < proba)

││││││ duréeTraitement = 2

│││││├── else

││││││ duréeTraitement = 1

│││││└──

││││└──

│││└──

││└──

│└──

└──────────

// supprimer le premier client de la file o─────────────o ↓ file

│ avancerFile │

o─────────────o ↓ file

┌─── \* **avancerFile**

│ iFile = 0

│ o────────────────────────o ↓ stations, file, iFile

│ │ copieClientVersStation │

│ o────────────────────────o ↓ stations

│ o─────────────────o ↓ iFile, file, longueurFile

│ │ supprimerClient │

│ o─────────────────o ↓ file, longueurFile

└──────────

o──────────────────o ↓ couts, nbStationsMin, nbStationsMax

│ rechercheCoutMin │

o──────────────────o ↓ nbStationsOptimal

┌─── \* **rechercheCoutMin**

│ iFinCout = nbStationsMax - nbStations

│ iCout = 0

│ coutMin = HV

│ iCoutMin = -1

│╔══ do while(iCout < iFinCout)

│║┌── if(couts[iCout] < coutMin)

│║│ iCoutMin = iCout

│║│ coutMin = couts[iCout]

│║└──

│║ iCout ++

│╙──

│ nbStationsOptimal = iCoutMin + nbStationsMin