Haute Ecole de Namur – Liège – Luxembourg
Département économique
Implantation IESN

Bachelier en Informatique de Gestion

Bloc 3

Recherche opérationnelle

Projet :

- Génération et validation par le test du carré-unité d'une suite de nombres pseudo-aléatoires
- À l'aide de cette suite, simulation d'un système de files d'attente

Professeurs : Isabelle CHARLIER, Corinne DERWA

Tinaël DEVRESSE

Romain SALE

Année académique 2018-2019

Énoncé

Votre programme aura pour but de

- permettre à l'utilisateur d'encoder les constantes a et c de la formule congruentielle linéaire mixte et un germe x_0, de tester la validité de cette suite par le test du carré-unité
- simuler un système de file d'attente sous certaines conditions et ce, si la suite est acceptée.

Le système d'attente se caractérise par

- une loi des arrivées des clients :

Arrivées par minute	0	1	2	3	4	5
Répétitions	16	20	12	5	3	2

- une loi des services :

Durée en minutes	1	2	3	4	5	6
Répétitions	18	21	15	3	1	1

Les hypothèses suivantes sont établies :

- Un client peut être prioritaire ou ordinaire. Il y a trois chances sur 10 pour qu'un client entrant soit prioritaire ;
- Deux files existent : une file avec les prioritaires et une autre avec les ordinaires ; leur capacité est illimitée ; les nouveaux arrivés se placent en file derrière les précédents ;
- Les deux premières stations sont réservées aux prioritaires ; ils ne peuvent se rendre dans les autres stations que si elles sont libres et qu'il n'y a pas d'ordinaires en file
- Le temps de simulation est de 960 minutes ;

Les coûts unitaires sont :

- pour une heure de présence dans le **système** : prioritaire : 40 euros, ordinaire : 25 ;
- pour une heure d'occupation d'une station : 35 ;
- pour une heure d'inoccupation d'une station : 20.

Déterminez le nombre optimal de stations à ouvrir.

En sorties,

- pour la première valeur envisagée du nombre de stations ainsi que pour la valeur 5,
  - pour chacune des 20 premières minutes,
    - le nombre d'arrivées ainsi que le type de chaque client;
    - en début de minute : les stations : par station occupée, type du client présent, durée de service restante ;
    - en début de minute : les files avec toutes les informations nécessaires pour en comprendre le fonctionnement ;
    - en fin de minute, les stations et les files pour bien visualiser ce qui s'est passé pendant la minute;
  - les différents coûts en fin de simulation ;
- pour les autres valeurs, les différents coûts en fin de simulation.

Le dossier à rendre au professeur concerné reprendra :

- l'énoncé reçu ;
- le diagramme d'actions (avec description des entrées-sorties et des structures créées ainsi que des modules avec noms adéquats, entrées/sorties des modules, …) ;
- si le DA a été accepté avec une note **d'au moins 14**/20, le programme écrit en langage C

Remarque : D'autres valeurs pour les coûts unitaires pourraient vous être fournies ultérieurement.

Descriptif des variables et structures

Variables

Nom	Type	Description	Constante
a	Int	Entier positif utilisé dans la formule linéaire congruentielle mixte	☐
c	Int	Entier positif utilisé dans la formule linéaire congruentielle mixte	☐
coûts	Double[]	Tableau de recensement des différents coûts au cours de la simulation	☐
d²	Double	Distance au carrée utilisée dans le test du carré-unité. Elle représente le carré de la distance entre deux points générés de façon pseudo-aléatoire	☐
duréeService	Int	Durée durant laquelle un client se trouve en station. Elle est générée de façon pseudo-aléatoire	☐
estValide	Bool	Caractère valide ou non de la suite pseudo-aléatoire générée avec les valeurs entrées par le client suite à la vérification par le test du carré-unité	☐
fileOrd	Int	Variable reprenant le nombre de clients ordinaires en file	☐
filePrio	Int	Variable reprenant le nombre de clients prioritaires en file	☐
iMinute	Int	Compteur de minute Valeur par défaut : 0	☐
iQueue	Int	Position courante de fin de file (Test carré-unité) Valeur par défaut : 20	☐
iTête	Int	Index du début de file (Test carré-unité)	☐
khi²Théoriques	Double[]	Tableau reprenant les valeurs de la table de khi² théoriques.	☐
khiCarré	Double	Valeur du khi² observé.	☐
m	Double	Entier positif utilisé dans la formule linéaire congruentielle mixte Valeur : HV + 1.0 avec HV = UINT_MAX	☐
n	Int	Nombre de valeurs générées par la suite Valeur par défaut : 10560	☐
nbArrivées	Int	Nombre d'arrivées sur une minute de simulation	☐
nbPremiere	Double	Première partie de l'addition de la formule de F(x)	☐
nbSeconde	Double	Deuxième partie de l'addition de la formule de F(x)	☐
nRépétitionsArrivéeParMinute	Int	Nombre maximum des répétitions cumulées de la loi des arrivées des clients	☐
nRépétitionsDuréeService	Int	Nombre maximum des répétitions cumulées de la loi des durées de service	☐

probabilités	Double[]	Tableau des probabilités (qu'une d^2 calculée se trouve dans tel ou tel intervalle)	☐
réel	Double	Réel entre 0 et 1 généré par la suite	☐
répétitionsD^2	Int[]	Tableau de répétitions (nombre de fois qu'on a déterminé une d^2 se trouvant dans tel intervalle)	☐
stations	Station[]	Tableau représentant l'état des différentes stations	☐
suite	Double[]	Tableau des valeurs de la suite pseudo-aléatoire	☐
x0	Int	Germe de la suite pseudo-aléatoire	☐
ALPHA	Float	Constante utilisée pour le tableau des khi^2 théoriques Valeur : 0.05	☑
M_PI	Double	Constante π de Math.h	☑
NB_STATIONS_MAX	Int	Nombre de stations maximum à ouvrir Valeur : 30	☑
NB_STATIONS_MIN	Int	Nombre de stations minimum à ouvrir Valeur = 4	☑
PRIX_MINUTE_INOCCUPATION	Double	Prix de l'inoccupation d'une station à la minute Valeur : 20/60	☑
PRIX_MINUTE_OCCUPATION	Double	Prix de l'occupation d'une station à la minute Valeur : 35/60	☑
PRIX_MINUTE_PRÉSENCE_ORD	Double	Prix de la présence d'un client ordinaire dans le système, à la minute Valeur : 25/60	☑
PRIX_MINUTE_PRÉSENCE_PRIO	Double	Prix de la présence d'un client prioritaire dans le système, à la minute Valeur : 40/60	☑
TEMPS_SIMULATION	Int	Temps de simulation en minutes Valeur : 960	☑

Structures

Station : { **clientPrio** (bool), **duréeServiceMax** (int), **duréeServiceRestante** (int), **numéroStation** (int) }

Enumération : { **iPrésencePrio** = 0, **iPrésenceOrd** = 1, **iOccupation** = 2, **iInoccupation** = 3 }

Autre

- Le sigle ^ symbolise l'utilisation de la fonction pow de la librairie Math.h
- Le texte acos() signifie l'utilisation de la fonction acos de la librairie Math.h
- Le texte sqrt() signifie l'utilisation de la fonction sqrt de la librairie Math.h

Diagramme d'actions

*** Génération des probabilités**

```
i = 0
x = 1
pbSeconde = 0
do while (i < 10)
 pbPremière = M_PI*(x/10) - (8/3)*(x/10)^(3/2) + (x/10)^2/2
 probabilités(x-1) = pbPremière – pbSeconde
 pbSeconde = pbPremière
 x++
 i++
do while (i ≤ 20)
 pbPremière = 1/3 + (M_PI-2)*(x/10) + 4*((x/10)-1)^(1/2) + (8/3)*((x/10)-
1)^(3/2) - (x/10)^2/2 - 4*(x/10)*acos(1/sqrt(x/10))
 probabilités(x-1) = pbPremière – pbSeconde
 pbSeconde = nbPremière
 x++
 i++
return probabilités
```

*** Initialisation des répétitions des distances² calculées**

```
i = iTête
do while (i < iQueue)
 répétitionsD²(i) = 0
 i++
return répétitionsD²
```

*** Génération d'un réel**

```
xn = (xn * a) +c
return xn / M, xn
```

*** Remplissage des répititions des distances² calculées**

```
d² = (suite(0) - suite(2))^2 + (suite(1) - suite(3))^2
répétitionsD²(ENTIER(d²*10))++
```

```
┌── * Validation
│ o──────────────────────────────────o ↓ degréLiberté=iQueue-1, ALPHA
│ │ Init tableau des khi² théoriques │
│ o──────────────────────────────────o ↓ khi²Théoriques
│ i = iQueue-1
│ ╔══ do while (i > iTête)
│ ╠══ if (probabilités(i)*N < 5)
│ ╠ probabilités(i-1) += probabilités(i)
│ ╠ répétitionsD²(i-1) += répétitionsD²(i)
│ ╠ iQueue--
│ ╠══
│ ╠ i--
│ ╚══
│ o──────────────────────────o ↓ iQueue, répititionsD², probabilités
│ │ Calcul khi²observable │
│ o──────────────────────────o ↓ khiCarré
│ return khiCarré ≤ khi²Théoriques(iQueue-1-1) // -1 parce qu'on respecte le
calcul et -1 parce que notre tableau commence à 0
└──────────
```

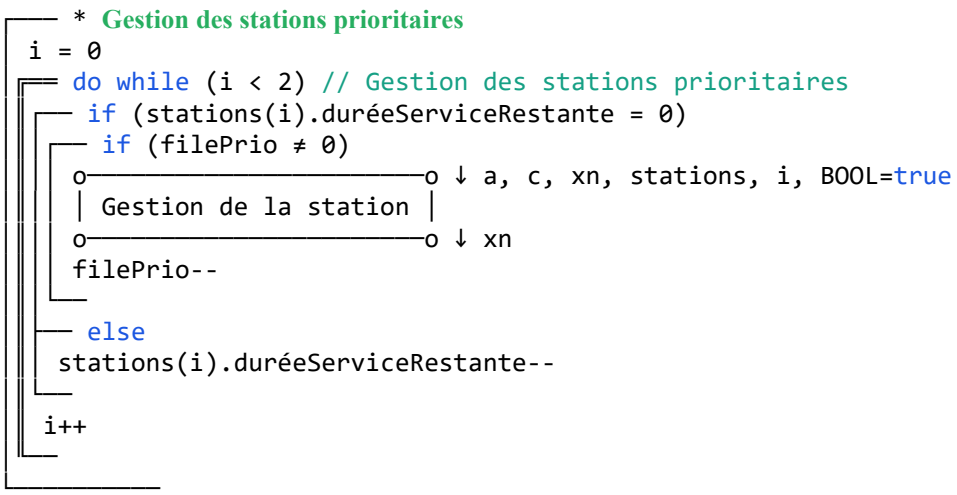
```
┌── * Calcul khi²observable
│ khiCarré = 0
│ i = 0
│ ╔══ do while (i < iQueue)
│ ╠ khiCarré += (répititionsD²(i) - N*probabilités(i))² / (N*probabilités(i))
│ ╠ i++
│ ╚══
│ return khiCarré
└──────────
```

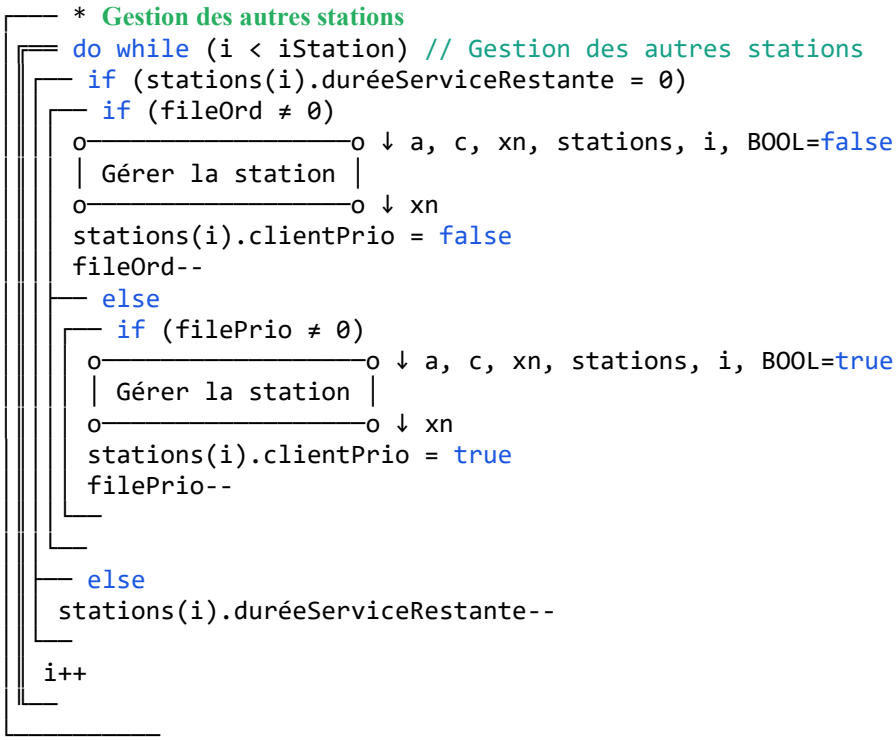
```
┌── * Système de files d'attente
│ optimum.cout = HV
│ optimum.nbStations = -1
│ iStation = NB_STATIONS_MIN-1
│╔═ do while (iStation < NB_STATIONS_MAX)
│ o──────o
│ │ Init │
│ o──────o ↓ coûts, filePrio, fileOrd, stations, xn
│ iMinute = 1
│╔═ do while (iMinute ≤ TEMPS_SIMULATION)
│ o──────────────────o ↓ a, c, xn
│ │ GénérerNbArrivées │
│ o──────────────────o ↓ nbArrivées
│ iArrivée = 0
│╔═ do while (iArrivée ≤ nbArrivées)
│ o──────────────────o ↓ a, c, xn
│ │ Génération Réel │
│ o──────────────────o ↓ réel, xn
│┌─ if (réel < 0.3)
│ filePrio++
│├─ else
│ fileOrd++
│└─
│ iArrivée++
│╚═
│ // impression demandée
│ o──────────────────────o ↓ iMinute, fileOrd, filePrio, coûts
│ │ Gestion de la minute │
│ o──────────────────────o ↓ coûts
│ // impression demandée
│ iMinute++
│╚═
│ // impression demandée
│ // recherche min (dans optimum.cout)
│ iStation++
│╚═
│ // impression optimum.nbStations
└──────────

┌── * Init
│ xn = x0
│ filePrio = 0
│ fileOrd = 0
│ iCoût = 0
│╔═ do while (iCoût < 4)
│ coûts(iCoût) = 0
│ iCoût++
│╚═
│ iStation = 0
│╔═ do while (iCoût < NB_STATIONS_MAX)
│ stations(iStation) ≡ 0 // REMETS TOUT A 0
│ iStation++
│╚═
│ return coûts, filePrio, fileOrd, xn
└──────────
```

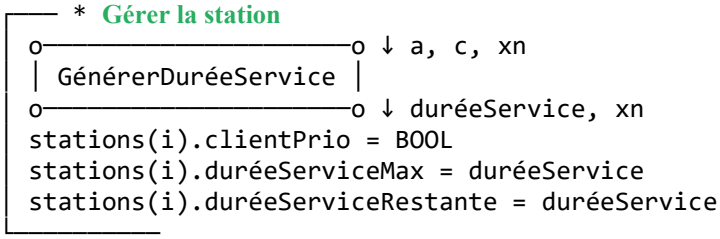
```
* Gestion de la minute
o──────────────────────────o ↓ iMinute, fileOrd, filePrio
│ Gestion de la simulation │
o──────────────────────────o ↓ xn
o────────────────────o ↓ coûts
│ Gestion des coûts  │
o────────────────────o ↓ coûts
return coûts
```

```
* Gestion de la simulation
┌──────────────────────────────────┐
│ Gestion des stations prioritaires │
└──────────────────────────────────┘
┌──────────────────────────────────┐
│ Gestion des autres stations      │
└──────────────────────────────────┘
```

```
* GénérerNbArrivées
o──────────────────o ↓ a, c, xn
│ Génération Réel  │
o──────────────────o ↓ réel, xn
réel *= 58 // nRépétitionsArrivéeParMinute
 if (réel < 16)
  nbArrivées = 0
 else
  if (réel < 36)
   nbArrivées = 1
  else
   if (réel < 48)
    nbArrivées = 2
   else
    if (réel < 53)
     nbArrivées = 3
    else
     if (réel < 56)
      nbArrivées = 4
     else
      nbArrivées = 5
return nbArrivées, xn
```


* Gestion des stations prioritaires
i = 0
do while (i < 2) // Gestion des stations prioritaires
if (stations(i).duréeServiceRestante = 0)
if (filePrio ≠ 0)
o↓ a, c, xn, stations, i, BOOL=true
Gestion de la station
o↓ xn
filePrio--
else
stations(i).duréeServiceRestante--
i++


* Gestion des autres stations
do while (i < iStation) // Gestion des autres stations
if (stations(i).duréeServiceRestante = 0)
if (fileOrd ≠ 0)
o↓ a, c, xn, stations, i, BOOL=false
Gérer la station
o↓ xn
stations(i).clientPrio = false
fileOrd--
else
if (filePrio ≠ 0)
o↓ a, c, xn, stations, i, BOOL=true
Gérer la station
o↓ xn
stations(i).clientPrio = true
filePrio--
else
stations(i).duréeServiceRestante--
i++


* Gérer la station
o↓ a, c, xn
GénérerDuréeService
o↓ duréeService, xn
stations(i).clientPrio = BOOL
stations(i).duréeServiceMax = duréeService
stations(i).duréeServiceRestante = duréeService

```
* GénérerDuréeService
o──────────────────o ↓ a, c, xn
 Génération Réel
o──────────────────o ↓ réel, xn
réel *= 59 // nRépétitionsDuréeService
if (réel < 18)
 duréeService = 1
else
 if (réel < 39)
  duréeService = 2
 else
  if (réel < 54)
   duréeService = 3
  else
   if (réel < 57)
    duréeService = 4
   else
    if (réel < 58)
     duréeService = 5
    else
     duréeService = 6
return duréeService
```

```
* Gestion des coûts
coûts(iPrésencePrio) += filePrio * PRIX_MINUTE_PRÉSENCE_PRIO
coûts(iPrésenceOrd) += fileOrd * PRIX_MINUTE_PRÉSENCE_ORD
iStation = 0
do while (iStation < NB_STATIONS_MAX)
 if (stations(iStation).duréeServiceRestance = 0)
  coûts(iInoccupation) += PRIX_MINUTE_INOCCUPATION
 else
  coûts(iOccupation) += PRIX_MINUTE_OCCUPATION
  if (stations(iStation).clientPrio)
   coûts(iPrésencePrio) += PRIX_MINUTE_PRÉSENCE_PRIO
  else
   coûts(iPrésenceOrd) += PRIX_MINUTE_PRÉSENCE_ORD
return coûts
```