Projet d'algorithmique

Cours d'Informatique de Deuxième Année

—Deug Sciences S3-MI—

Simulation de files d'attente d'un grand magasin

Le but du projet est d'écrire un programme qui simule le passage en caisse des clients d'un grand magasin.

- Ce travail doit être rendu aux chargés de TD ou TP, ou au plus tard le vendredi 7 janvier 2000 à 12h45 à Bruno Gauthier (salle 10 bâtiment Clément Ader) ou à Jocelyn Marchadier (salle 20 bâtiment Lavoisier). **Vous indiquerez les deux noms du binôme ainsi que le nom de votre chargé de TD et celui de votre chargé de TP** ;
- le dernier TP (semaine du 10 au 14 janvier) est annulé et remplacé par une séance de soutenance qui aura lieu le même jour pour tout les groupes de TP (entre le 10 et le 14 janvier). La date de cette séance vous sera précisée ultérieurement ;
- la clarté et le soin apportés à la rédaction de votre travail, ainsi que la lisibilité de votre *listing* seront pris en compte ;
- ce travail peut être réalisé en binôme, chaque personne devant être présente à la soutenance et pouvant être interrogée individuellement ;
- la partie programmation devra être réalisée en *C ANSI*, compilable directement avec `gcc` sous Linux.

---

1 Présentation du sujet

Le projet de ce semestre consistera en la réalisation d'un logiciel de simulation de passage en caisse des clients d'un grand magasin. Tout le monde a, au moins une fois dans sa vie, fait des achats dans un des nombreux hypermarchés de France. Pour un client, le temps de passage en caisse est — dans ce mode de distribution — une part non négligeable du temps passé dans le magasin (du point de vue commercial, il ne dépense rien pendant ce labs de de temps — excepté un éventuel paquet de chewing-gum). Il est donc économiquement judicieux de s'intéresser à cette étape du parcours du parfait consommateur.

2 Modélisation

Dans cette section, nous définissons un *modèle* pour simuler le passage en caisse des clients. Nous allons considérer un magasin comme un système évoluant au cours du temps. Nous suivrons cette évolution selon les contraintes que nous imposerons.

Définir un modèle, c'est se donner un ensemble d'objets permettant de décrire l'état de notre système à un instant donné. Ici les différents objets sont les clients, les files d'attente et les caisses.

Afin de simplifier le problème, nous proposons un modèle de simulation *à temps discret*. On observe donc l'évolution du système à intervalle de temps régulier (chaque minute par exemple). À un instant donné, plusieurs événements peuvent modifier l'état du système. Parmi ceux-ci, on peut — à titre d'exemple — distinguer l'arrivée d'un client, la fin de passage en caisse d'un client, l'ouverture ou fermeture d'une caisse, ...

Le système évolue également de façon automatique (entre deux instants successifs). Les déplacements des clients de file en file sont générés afin de minimiser les temps d'attente aux caisses.

3 Choix des structures de données

Nous considérons un magasin qui possède un nombre `N` de caisses que nous pouvons alors représenter par un tableau statique. Chaque caisse possède plusieurs attributs (ouvert/fermé, file d'attente, ...). On pourra aussi stocker le temps d'attente total de la file afin d'éviter de le recalculer à chaque fois que l'on en a besoin.

Une file d'attente contient les informations de chaque client qui la composent (leur nom et leur durée de passage en caisse). On peut se poser la question de savoir s'il est judicieux de stocker le temps d'attente de chaque client (avec mise à jour régulière) ou bien de le recalculer au besoin.

Le choix de l'implantation de la file d'attente dépend des opérations que l'on veut effectuer sur cette file (insertion en queue, suppression en queue, ...). À vous de faire le meilleur choix parmi les listes chaînées, doublement chaînées, ...

4 Étude des déplacements

☞ À chaque instant, il vous faudra décider des mouvements des clients qui peuvent bénéficier de temps d'attente moins longs dans une autre file. La question revient à chercher si, pour un client donné, une caisse voisine n'est pas moins "chargée". Vous devrez expliquer quel algorithme vous utilisez pour décider ces changements de caisse. La méthode utilisée influencera probablement le raffinement des structures de données. Vous devrez préciser vos choix.

5 Description type d'une session

5.1 Quelques généralités

Le programme que vous devez faire doit être interactif, au sens où l'utilisateur doit pouvoir modifier le système. Nous imposons les contraintes suivantes :

- un client peut continuer ses achats pendant qu'il attend à une caisse (on peut imaginer que son accompagnateur fait quelques derniers achats). La durée de passage en caisse de ce client peut donc évoluer.

- un client peut se déplacer uniquement sur les deux caisses voisines vers la droite et vers la gauche (on considère que la mauvaise visibilité l'empêche d'apprécier correctement le temps d'attente aux autres caisses) ;

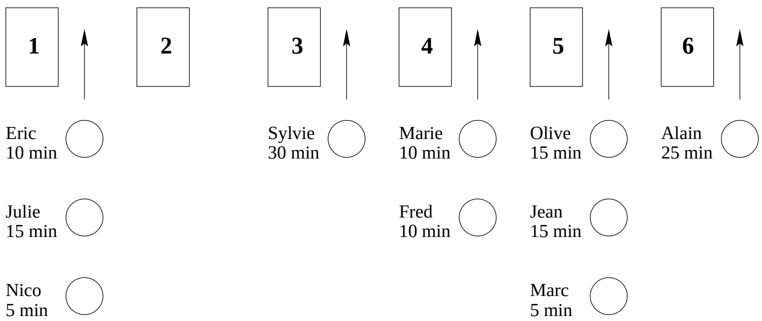
- il n'y a pas de resquilleurs dans notre modèle.

On pourra prévoir un petit langage de commandes permettant à l'utilisateur de passer ses demandes au programme. Le langage utilisé devra être très simple et pourra s'inspirer de l'exemple suivant.

Au lancement de `a.out` (par exemple), le programme affiche un *prompt*—nous avons choisi le *prompt* `MAG>`—et se met en attente d'un ordre de la part de l'utilisateur. La première commande utile sera la commande `h` fournissant de l'aide sur les commandes disponibles. L'utilisateur peut désormais demander le chargement d'une "situation de départ" par la commande `c` à partir d'un document de nom `fichier.txt`. L'utilisateur peut également demander des modifications sur la configuration (un client change de caisse, un client "lassé d'attendre" part sans rien acheter, ...).
Les événements importants (notamment les mouvements et les départs des clients) devront être indiqués à l'affichage.

5.2 Exemple de session

On considère la configuration suivante,



qui est codée dans un fichier texte par le formalisme suivant :
`Numéro_de_caisse Nom_Client Durée_Client`
(la file de la caisse 1, puis celle de la caisse 2, ...).

```
$ cat fichier.txt
1 Eric 10
1 Julie 15
1 Nico 5
3 Sylvie 30
4 Marie 10
4 Fred 10
```

```
5 Olive 15
5 Jean 15
5 Marc 5
6 Alain 25
$ a.out
MAG> h
```

Commande	*Effet*
`q`	Pour quitter le programme
`h`	Pour obtenir cette aide
`c nom_de_fichier`	Chargement d'une "situation"
`s nom_de_fichier`	Sauvegarde d'une "situation"
`a numéro`	Affichage des informations relatives aux clients qui attendent à la caisse `numéro`
`t [n]`	Passage du temps de t à $t+n$ (n sauts de 1). (si `n` n'est pas précisé, le saut est de 1.)
`o numéro`	Ouverture de la caisse `numéro`
`f numéro`	Fermeture de la caisse `numéro`
`n numéro client durée`	Arrivée d'un nouveau `client` (dont la durée de passage en caisse est `durée` vers la caisse `numéro`)
`m client num1 num2`	Mouvement du client de la caisse `num1` vers la caisse `num2`
`d client durée`	Modification de la durée de passage du client `client`

```
MAG> c fichier.txt
MAG> a 3
Sylvie (30min)
MAG> a 5
Olive (15min)
Jean (15min)
Marc (5min)
MAG> o 2
MAG> n 2 Frederique 10
MAG> n 2 Nicole 5
MAG> a 2
Frederique (10min)
Nicole (5min)
MAG> t
Nico s'est deplace de la caisse 1 vers la caisse 2
Marc s'est deplace de la caisse 5 vers la caisse 4
MAG> a 2
Frederique (9min)
Nicole (5min)
Nico (5min)
MAG> a 4
Marie (9min)
Fred (10min)
Marc (5min)
```

6 Travail à rendre

Vous devrez fournir un rapport qui comprendra les points suivants : un cahier des charges, une présentation du modèle, une description et une justification des algorithmes et des structures de données.

Naturellement, vous devrez employer des structures de données claires et efficaces associées à des algorithmes aussi performants que possibles dont vous évaluerez la complexité.

Il est très important que le code soit clair et *judicieusement* commenté. Il est plus important d'avoir un projet bien documenté avec un source bien commenté, de bons algorithmes que de réaliser des options.

Pour la soutenance, vous devrez prévoir au moins deux fichiers tests judicieusement choisis tels que vous puissiez mettre en lumière l'*intelligence* de votre programme.

N'OUBLIEZ PAS : VOTRE PROGRAMME DEVRA "PASSER L'AN 2000".

7 Améliorations et options

Les améliorations et les options ne sont pas obligatoires, mais constituent un *plus* pour l'évaluation finale de votre travail. Vous êtes invités à faire part de vos propositions aux enseignants pour savoir dans quelle mesure elles sont réellement pertinentes.
On peut envisager les options suivantes :

- l'émergence d'événements aléatoires (panne d'une caisse pendant 5 minutes, changement de caissière, ...) ;
- la création d'une caisse réservée aux petits acheteurs (durée $\leq$ 5 minutes) ;
- l'édition de statistiques sur les temps d'attente ;
- la présence de resquilleurs (qui peuvent se déplacer où ils veulent). Par exemple :
  `r client durée numéro pos` correspond à l'arrivée d'un resquilleur (caractérisé par son nom `client` et sa durée de passage en caisse `durée`) dans la file de la caisse `numéro` en position `pos` (où `pos` est compris entre 0 (tête de la file) et n (queue de la file) ;
- le principe de courtoisie qui consiste à donner le passage à un client dont la durée de passage est négligeable devant un autre client.

Nous vous laissons la plus grande liberté dans les options que vous voudrez ajouter. Cependant, vous devrez justifier d'un point de vue algorithmique vos améliorations.