EXPLICATIONS PROJET C

I- Organisation du programme

Le projet est séparé en trois fichiers :

- projet_c.c contenant toutes les fonctions du projet
- projet_c.h contenant tous les prototypes de ces fonctions
- main.c contenant la fonction main
- makefile permettant la compilation

Pour compiler, écrire dans la console « make projet » puis écrire « ./projet » pour exécuter la fonction. Une fois compilée et exécutée, vous devez rentrer alpha, lambda, le cycle du feu et la durée souhaitée de la simulation en secondes ou en nombre de cycles (au choix). Le programme s'exécute et, une fois fini, un message apparaît dans la console et vous informe que toutes les informations concernant les voitures et les performances du feu ont été écrites dans des fichiers textes situés dans le dossier des fonctions.

On a interprété la valeur alpha comme étant la durée que met une voiture à avancer d'une case quand elle est à l'arrêt. Donc si la voiture est à l'arrêt à la 6ème place dans la file, elle mettra 6*alpha pour atteindre le feu (s'il ne passe pas au rouge entre temps).

Le fichier contient 2 fichiers « .c » et un fichier « .h ». Le premier fichier « .c » intitulé « projet_c.c » contient toutes les fonctions tandis que le fichier « main.c » contient uniquement la fonction main. Le fichier « projet_c.h » contient lui les prototypes de toutes les fonctions utilisées.

Le fichier « projet_c .c » est lui-même séparé en 8 parties :

- Calcul du temps d'arrivée
- Calcul du temps de passage
- Calcul de la durée d'attente
- Ajustements
- Ecriture fichier
- Lecture fichier
- Tableau de bord
- Fin

Où l'on retrouve dans chaque partie la fonction correspondant au titre ainsi que toutes les fonctions auxiliaires dont elle a besoin pour fonctionner.

Pour effectuer nos tests, on a pris :

- Alpha = 3 secondes
- Lambda = 0.1
- Durée du feu vert = durée du feu rouge = 30 secondes
- Durée du feu orange = 6 secondes

`

II-Structure des données

Nous avons choisi d'utiliser une liste chaînée pour représenter la file des voitures ainsi que la structure suivante pour représenter les voitures :

```
typedef struct _Voiture
{
        int num_v;
        int ta;
        int tp;
        int da;
        struct _Voiture *suiv;
} Voiture;
```

Où :

- num_v contient le numéro de la voiture dans l'ordre de leur arrivée
- ta contient le temps d'arrivée de la voiture
- tp contient le temps de passage de la voiture
- da contient la durée d'attente de la voiture
- suiv pointe vers la voiture qui arrive après elle.

Nous avons également créé deux autres structures pour représenter un cycle de feu et pour pointer sur le premier élément de la liste des voitures :

```
typedef struct
{
        Voiture *tete;
} Liste_voitures;


typedef struct
{
        int fv;
        int fo;
        int fr;
} Cycle;
```

III-Fonctions

Toutes les fonctions sont annotées dans les fichiers .c, mais pour mieux comprendre ces dernières, nos raisonnements seront décrits ci-dessous.

III-1) Temps d'arrivée

Comme le temps d'arrivée de la 3ème voiture dépend du temps d'arrivée de la 2ème voiture, il a fallu qu'on initialise la 1ère voiture. Pour ce faire, on l'a placée devant le feu à t=0, donc à ta (=temps d'arrivée) = 0. Ainsi, le temps d'arrivée de la 2ème voiture sera le temps d'arrivée de la 1ère voiture + X.

```c
void temps_arrivee(int T, Liste_voitures *l, float lambda) //T représente le temps total de l'expérience
{
	/*Il faut d'abord créer la liste et initialiser la première voiture*/
	Voiture *voit;
	voit = (Voiture*)malloc(sizeof(Voiture));
	voit->suiv = NULL;
	voit->num_v = 1;
	voit->ta = 0;
	voit->tp = 0;
	voit-> da = 0;

	l->tete = voit;		//La première voiture est mise en début de liste

	int X;		//X contient le temps entre deux voitures
	float U;
	int compteur = 0;
	int compteur_voiture=2;

	while (compteur<T)	//Tant qu'on n'a pas dépassé le temps voulu, on ajoute une voiture
	{
		U = (float)rand()/(float)RAND_MAX; //U e [0;1]
		//printf("U=%f\n",U);
		X = floorf(-logf(1-U)/lambda);		//X suit une loi exponentielle et représente le temps entre deux voitures

		//printf("X=%f\n",X);
		ajout_voit_tps_arrivee(l,X,compteur_voiture);	//On ajoute enfin la voiture en fin de liste tout en initialisant son temps d'arrivée et son numéro dans la liste.
		compteur += X;
		compteur_voiture += 1;
	}
}
```

Pour former la liste, on insère chaque nouvelle voiture à la fin de cette dernière grâce à la fonction « ajout_voit_tps_arrivee » qui ajoute la voiture en initialisant son numéro dans la file et son temps d'arrivée (notamment grâce à la fonction « insertion_fin »).

III-2) Temps de passage

C'est la fonction la plus complexe (c'est pourquoi toutes les étapes sont décomposées et annotées dans le fichier .c). Comme son nom l'indique, elle initialise le temps de passage de chaque voiture.

Il faut différencier plusieurs cas :

- Le cas où le feu est vert et il n'y a pas de file d'attente ; dans ce cas, la voiture ne s'arrête jamais et a un temps de passage égal au temps d'arrivée. (C'est le cas le plus simple, géré ligne 131-137)
- Le cas où le feu est orange ou rouge ; je vais construire la « liste » des voitures qui s'arrêteront pendant ce temps (géré ligne 165-175). (Cas plus complexe, géré ligne 140-208)

  Ici, le terme liste est employé mais on utilise en réalité un pointeur prem_voit qui pointe sur la première voiture arrêtée et un pointeur derniere_voit qui point sur la dernière voiture arrêtée.

  On a encore une fois plusieurs cas à distinguer :
  - Lors du prochain feu vert, toutes les voitures passeront et la liste se videra (ligne 181-192)
  - La file ne se videra pas, toutes les voitures ne passeront pas au feu vert.

  Dans ce dernier cas, il faut alors que les voitures attendent un voire plusieurs cycles avant de franchir le feu (et donc de quitter la liste). Cela correspond aux lignes 196-206.

On crée une fonction « ajout_cycle_voit » qui ajoute la durée d'un cycle au temps de passage de toutes les voitures de la « liste »

Pour bien ajouter le temps que met la voiture à passer le feu quand elle est arrêtée (dépendant donc de sa position dans la file d'arrêt), on modifie le numéro (num_v) de la voiture pour qu'il corresponde à sa position dans la file d'arrêt. (Si la voiture a le temps de passer pendant le prochain feu vert, on incrémentera alors son temps de passage de alpha*son numéro)

III-3) Durée d'attente

Cette fonction initialise la durée d'attente de toutes les voitures. On a considéré que la durée d'attente était la différence entre le temps de passage et le temps d'arrivée.

III-4) Lecture/Écriture du fichier

On a choisi d'écrire dans un fichier texte bien qu'il prenne plus de place qu'un fichier binaire pour sa lisibilité et pour nous permettre de vérifier nos résultats. Le fichier est organisé en 4 colonnes :

- Numéro de la voiture
- Temps d'arrivée
- Temps de passage
- Durée d'attente

Le fichier est écrit à partir d'une liste de voitures et est lu en insérant toutes les données lues dans une liste de voitures (où chaque voiture est bien-entendu insérée à la fin de cette dernière).

On a également écrit dans un fichier texte appelé « tableau_de_bord.txt » les performances du feu.

III-5) Interface

On a créé une fonction « appli » qui permet de lancer la simulation (donc appelée dans la fonction *main*) et où l'on doit rentrer alpha, lambda etc.