REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
ECOLE NATIONALE SUPERIEURE D'INFORMATIQUE

CPI 1 **Année 2017 / 2018**

TP
EN ALGORITHMIQUE ET STRUCTURES DE DONNEES DYNAMIQUES(ALSDD)

REALISE PAR:
MECHOUAR FELLA
BEDDEK LILYA

Section : C

Groupe N° : 10

Semestre 2

Table des matières

1. Les structures utilisées :

A) Définition des types :

```
Type Chambre = structure
Num_chambre :entier
Type:entier
Statut :entier
Datedeb :chaine
Datefin :chaine
Nom :chaine
Prix :entier
Ch_suiv : pointeur( chambre)
    Fin
Type hotel = structure
Nomhotel :chaine
Ville :chaine
Capacite :entier
Nb_chlib :entier
Nb_chocc :entier
Prix_hotel1 :entier
Prix_hotel2 :entier
Tete_chocc : pointeur (chambre)
Tete_chlib :pointeur (chambre)
Hot_suiv :poineur (hotel)
    Fin
```

```
Type lettre = structure
    Letr :caractere
    Hot : pointeur (hotel)
    Let_suiv :pointeur (lettre)
    Fin
```

B) Présentation du contenu des 3 structures :

Nous allons procéder en commençant par la dernière structure pur arriver à la dernière ;

La structure de la 3ème liste (chambres) :

- Les variables (num_chambre, type, statut, prix) de type entier designent succissivement le numero de chambre, le type (individuelle ou a deux), le statut qui décrit l'état de la chambre (1) si elle est occupée et (0) si elle est libre et finalement le prix qui indique le prix de la chambre qui est commun entre toutes les chambres de même type dans un meme hôtel.
- les variables de type caractère designent à leur tour :
  Datedeb : la date de début de la réservation

  Datefin : la date fin da réservation

  Nom : le nom du client

- Le champ ch_suiv disigne : le champ aderesse du maillon de la troisième liste qui indique l'adresse de la prochaine chambre.

La structure de la 2ème liste (liste des hôtels) :

- Les variable de type entier suivantes designent :
  Capacité : le nombre total des chambres que contient cet hôtel en question.
  Nb_chlib : le nombre de chambres libres (non reservées) dans cet hôtel.
  Nb_chocc : le nombre de chambres occuppées (reservées) dans cet hôtel.
  Prix_hotel1 : le prix d'une chambre de type individuelle dans cet hôtel.
  Prix_hotel2 : le prix d'une chambre de type à deux dans cet hôtel.
- Les variables de type chaine designent à leurs tour :
  Nomhotel : le nom de l'hôtel.

  Ville : la ville où est situé cet hôtel.

- Les trois pointeurs :
  Tete_chocc : pointeur vers la liste des chambres occuppées de cet hôtel.

  Tete_chlib : pointeur vers la liste des chambres libres de cet hôtel.

  Hot_suiv : le champ adresse du maillon de la deusième liste qui indique l'adresse du prochain hôtel de cette liste.

La structure de la 1ère liste (annuaire) :

- La variable letr de type caractère désigne une lettre de l'alphabet
  hot : pointeur vers la liste des hotels commançant par cette lettre.
  Let_suiv : le champ adresse du maillon de la première liste qui indique l'adresse du prochain maillon qui contient la prochaine lettre alphabetique.
  En ce qui concerne cette liste nous avons choisi de travailler avec une liste annuaire fixe de 26 maillons sans suppression d'aucun maillon même s'il ne contient aucun hôtel, et dans ce cas son champ hot contiendra NULL.

Pour une utilisation plus facile des listes nous avons utilisé une machine abstraite pour chaque type de liste qui caractérise les fonctionnalités élémentaires possibles.

C) Les machines abstraites :

- Machine abstraite de la liste des chambres :

```
/*************************************************************************************************************/
struct chambre* allouer_chambre();
/*************************************************************************************************************/
void affadr_chambre (struct chambre* p ,struct chambre* q);
/*************************************************************************************************************/
struct chambre* suivant_chambre (struct chambre* p);
/*************************************************************************************************************/
void aff_nb_chambre (struct chambre* p,int numch);
/*************************************************************************************************************/
int num_chambre(struct chambre* p);
/*************************************************************************************************************/
char* datedeb(struct chambre* p);
/*************************************************************************************************************/
char* datefin(struct chambre* p);
/*************************************************************************************************************/
void aff_datedeb (struct chambre* p, char datedeb[11]);
/*************************************************************************************************************/
void aff_datefin (struct chambre* p ,char datefin[11]);
/*************************************************************************************************************/
void aff_typechambre (struct chambre* p,int typech );
/*************************************************************************************************************/
int type_chambre (struct chambre* p);
/*************************************************************************************************************/
void aff_statutchambre (struct chambre* p, int statutch);
/*************************************************************************************************************/
int statut_chambre(struct chambre* p);
/*************************************************************************************************************/
```

```
/*************************************************************************************************************/
int statut_chambre(struct chambre* p);
/*************************************************************************************************************/
void aff_prixchambre (struct chambre* p ,int prixch);
/*************************************************************************************************************/
int prix_chambre(struct chambre* p);
/*************************************************************************************************************/
 char* nom_client(struct chambre* p);
/*************************************************************************************************************/
void aff_nomclient (struct chambre* p,char nomc[26]);
/*************************************************************************************************************/
```

- Machine abstraite de la liste des hôtels :

```
/*************************************************************************************************************/
struct hotel* allouer_hotel();
/*************************************************************************************************************/
void affadr_hotel (struct hotel* p,struct hotel* q);
/*************************************************************************************************************/
struct hotel* suivant_hotel(struct hotel* p);
/*************************************************************************************************************/
char* nom_hotel(struct hotel* p);
/*************************************************************************************************************/
char* ville(struct hotel* p);
/*************************************************************************************************************/
void aff_nomhotel(struct hotel* p,char nomhot[15]);
/*************************************************************************************************************/
void aff_nomville(struct hotel* p,char nomville[15]);
/*************************************************************************************************************/
void aff_nbchambres (struct hotel* p, int nb_chambres);
/*************************************************************************************************************/
int nb_chambres (struct hotel* p);
/*************************************************************************************************************/
void aff_nbchlibres (struct hotel* p , int nb_chamblibres);
/*************************************************************************************************************/
int nb_chambresvides(struct hotel* p);
/*************************************************************************************************************/
void aff_nbchoccupes (struct hotel* p,int nb_choccupes );
/*************************************************************************************************************/
int nb_choccupes (struct hotel * p);
/*************************************************************************************************************/
```

```
/*****************************************************************************************************/
int nb_chambres (struct hotel* p);
/*****************************************************************************************************/
void aff_adr_chambresocc(struct hotel* p,struct chambre* q);
/*****************************************************************************************************/
void aff_adr_chambreslib(struct hotel* p,struct chambre* q);
/*****************************************************************************************************/
struct chambre* adr_tetechocc (struct hotel* p);
/*****************************************************************************************************/
struct chambre* adr_tetechlib (struct hotel* p);
/*****************************************************************************************************/
void aff_prixhotel1(struct hotel* p,int prix);
/*****************************************************************************************************/
int prix_hotel1(struct hotel* p);
/*****************************************************************************************************/
void aff_prixhotel2(struct hotel* p,int prix);
/*****************************************************************************************************/
int prix_hotel2(struct hotel* p);
```

- Machine abstraite de l'annuaire :

```
/*****************************************************************************************************/
struct lettre* allouer_lettre ();
/*****************************************************************************************************/
void affadr_lettre ( struct lettre* p,struct lettre* suivant);
/*****************************************************************************************************/
void aff_lettre(struct lettre* p,char lettre);
/*****************************************************************************************************/
char lettre (struct lettre* p);
/*****************************************************************************************************/
struct lettre* suivant_lettre (struct lettre* p);
/*****************************************************************************************************/
void aff_adrtetehotel (struct lettre* p,struct hotel* tetehotel);
/*****************************************************************************************************/
struct hotel* adr_tetehotel(struct lettre* p);
/*****************************************************************************************************/
```

2. *Découpage modulaire :*

Pour le menu imposé

- Modules de fichiers :
  On a utilisé pour cela 4 modules, 2 pour la liste des hôtels et 2 pour la liste des réservations ;
  1- Pour le chargement des informations depuis le fichier texte.
  2- Pour la sauvegarde.
- Affichage du statut des réservations : à travers le parcourt des 3 listes imbriquées, ce module affiche toutes les réservations (les champs des chambres occupées) pour chaque hôtel présent sur nos listes.
- Insérer réservation : donne la possibilité à l'utilisateur d'insérer une nouvelle réservation dans l'hôtel de son choix, après avoir introduit le nom de l'hôtel, sa situation, et le numéro de la chambre. Si ce module est présenté indépendamment des autres (choix 3 du menu), on lui présentera avant toutes les listes des chambres libres existantes dans chaque hôtel.
- Modifier réservation : ce dernier nous donne la possibilité de modifier une réservation selon 6 choix:
  1/ Les dates de réservation (début ou fin, ou les deux), en respectant bien évidement quelques règles fixées par un autre module dont on parlera par la suite.
  2/ Le nom du client sous lequel la réservation a été enregistrée.

3/ Le type de chambre, en lui affichant toutes les chambres du type opposé pour qu'il en choisisse une et qu'il l'a réservé en utilisant le module précédant et en prenant soin de supprimer la réservation initiale.
4/ Son numéro de chambre si par exemple le client n'a pas aimé son emplacement.
5/ Son hôtel, si la possibilité d'un hôtel plus intéressant s'offre à lui.
6/ Changer de ville en suivant le même concept de suppression d'une réservation et de l'insertion d'une nouvelle, en ne prenant en compte cette fois ci que le nom de la nouvelle ville.

- Supprimer une réservation : cette fonctionnalité sert à supprimer une réservation en précisant le nom du client seulement.
- Rechercher une chambre libre selon son type et selon la demande du client :
    1. Par ville : en lui donnant la main pour introduire le nom de la ville qu'il désire.
    2. Par hôtel : si le client vise un hôtel précis.
    3. Ayant le plus petit prix : si le client cherche la chambre la moins chère.
- Rechercher la réservation effectuée par une personne « Nom » : juste en introduisant le nom de la personne.
- Rajouter un hôtel : on offre la possibilité de rajouter un hôtel à condition de son absence dans nos listes (par nom et ville), si cette condition est vérifiée, il pourra compléter par le reste des informations concernant l'hôtel.
- Supprimer un hôtel : en précisant le nom et la ville de l'hôtel, il sera supprimé.
- Libérer toutes les chambres occupées ayant une date de fin égale à la date du jour : en prenant en considération à chaque exécution la date du système, les réservations ayant la date fin égale à la date du jour seront automatiquement supprimées.
- Trier les listes des hôtels par ordre alphabétique : cette fonctionnalité sera réalisée en utilisant le trie par bulles en plus du trie par insertion (chaque nouvel hôtel inséré sera introduit dans la liste selon un chainage précis respectant l'ordre alphabétique).

  Des foncions secondaires seront rajoutées à ces fonctionnalités pour permettre d'afficher la majorité des messages d'erreurs possibles (par exemple, les intervalles des jours et des mois, introduction d'une date fin précédant la date début…etc.).

- Les entêtes de modules :

```
/*****************************************
            LES FICHIERS
*****************************************/

void creer_listlettre (struct lettre** tetelettre);
void creer_listchambreslibres (struct lettre* tete);
void afficher_listchambresvides (struct lettre* tetelettre);
void afficher_listhotel (struct lettre* teteletrre);
void creer_listechabreocc(struct lettre * tetelettre);
void afficher_listchambresocc(struct lettre* tetelettre);
void sauvegarder_listhotel(struct lettre* tete);
void sauvegarder_resrvation (struct lettre* tete);
void sauvegarder_chambresvides (struct lettre* tete);
/*****************************************/
```

```
/*****************************************
      PROBLEME DE DATES
*****************************************/
int cmp_df(char datedebut[],char datef[]);
int verif_date(char newdatedeb[],char newdatef[]);
int verif_date_fin(char newdatef[]);
```

```
/*****************************************************************************************************************/
                    /*********************** LES MODULES DU MENU ****************************/
/*****************************************************************************************************************/
void inserer_maillon(struct chambre** t, struct chambre* r);
void supprimer_maillon(struct chambre** t,struct chambre* s,struct chambre* r);
void supprimer_reservation(struct lettre* t, char nomclient[]);
void liberer_list(struct chambre** t);
void supprimer_hotel(struct lettre* t, char nomhotel[],char vil[]);
void plus_bat_prix(struct lettre* t,int type);
int recherche_ville(struct lettre* t, char vil[],int type);
int recherche_hotel(struct lettre* t,int type,char nom[15]);
void afficher_statut_reserv(struct lettre* t);
void inserer_reservation(struct lettre* t,char nomhotel[],char nom[],char vil[],int numch,char datedeb[],char datefin[]);
int recherche_tab_ville(char* tab[48],int tai,char ville[15]);
void toutes_nos_villes(struct lettre* t);
void modifier_reservation(struct lettre* t, char nomclient[26]);
void inserer_hotel(struct hotel** t,struct hotel* r);
struct chambre* creer_liste_cblibre(int nb_chambre,int prixa1, int prixa2);
void introduire_nouvel_hotel(struct lettre* t,char nomhotel[],char nomville[],int nbchambre,int prixa1,int prixa2);
void tri_bulles(struct hotel** t);
void date_fini(struct lettre* t);
void recherche_reservetion(struct lettre* t,char nomclient[]);
void libe_tout(struct lettre** t);
```

3. Manuel d'utilisation :

Pour manipuler nos listes plus facilement et pour éviter de perdre quelconque information nous avons choisi de travailler avec trois fichiers :

Un fichier « hotel.txt » pour la liste des hôtels, qui contient toutes les informations concernant chaque hôtel que nous avons et qui servira à charger la structure hôtel a chaque exécution du programme (appel à l'application)

Un fichier« reservation.txt » qui contient toutes les informations sur n'importe quelle réservation effectuée dans un hôtel que nous avons et qui servira à remplir la liste des réservations et qui constitue au même temps le premier choix du menu, autrement tout le travail que vous ferez s'effectuera sur des hôtels ne contenant aucune réservation.

un fichier contenant les listes de chambres vides de chaque hôtel pour ne pas perdre leurs numéros entre les différentes exécutions, vu que les numéros de chambres occupées ne se suivent pas forcement et si à chaque exécution on allouait des maillons en leur affectant des numéros aléatoires, on pourrait tomber sur le cas d'une chambre qui serait occupée et libre au même temps, sauf pour le cas d'un nouvel hôtel inséré à qui on allouera une liste de chambres numérotées de 1 jusqu'au nombre de chambre totales et qui seront toutes libres.

Notre application offre 12 fonctionnalités et la 13ème pour quitter le programme définitivement, et après chaque exécution d'un choix ; la main sera donnée pour cliquer sur n'importe quelle touche et revenir au menu principal.

Avant de quitter le programme vérifiez bien que vous avez sauvegardé tout ce que vous avez fait en choisissant le numéro 12 pour pouvoir charger tout ce que contiennent les listes (hôtels ou réservation) dans de nouveaux fichiers (newhotel.txt newreservation.txt newchambresvides.txt) qui préserveront toutes les modifications que vous aurez effectué sur les anciennes listes.

Si vous voulez relancer l'application et travailler avec les listes modifiées, veuillez supprimer les anciens fichiers (hotel.txt, reservation.txt, chambresvides.txt) et renommer les nouveaux fichiers en enlevant le « new » de chacun d'eux pour avoir les mêmes noms que les précédents.

Concernant les autres fonctionnalités du menu, vous aurez juste à introduire le numéro du choix en question et suivre les étapes écrites pour l'exécuter.

Quelques remarques pour les différentes fonctionnalités :

Si vous voulez afficher le statut des réservations et que vous n'avez pas chargé les réservations (choix un), vous ne verrez aucune réservation s'afficher sauf celles que vous aurez introduites depuis le lancement de l'application.

Pour toutes les fonctionnalités dans lesquelles vous aurez à faire entrer un nom du client (insertion ou modification d'une réservation.), et si vous avez des noms complexes ou bien vous souhaitez introduire vos noms et prénoms au même temps, soyez sûrs de les séparer par un tiret ou n'importe quel autre caractère, ne laissez pas de vide (blanc) entre les mots.

Pour les modules de recherche de chambres libres (en introduisant le nom de la ville, l'hôtel ou même en cherchant celle ayant le plus bas prix vous verrez s'afficher devant vous toutes les chambres libres trouvées en fonction de la demande émise, mais sans faire de réservation, alors, vous pouvez compléter votre recherche par une réservation, retenez ses coordonnées (nom de ville nom de l'hôtel numéro de chambre) revenez au menu et choisissez la fonctionnalité qui vous permet de l'insérer.

Si vous décidez d'insérer une réservation, et que cette opération de déroule avec succès, un message apparaitra vous permettant d'enregistrant votre demande dans un fichier texte, cette fonctionnalité vous permet de créer un fichier texte contenant toutes les informations liées à votre réservation, il se trouvera dans le même dossier du projet et il sera nommé mareservation.txt.

Si vous décidez d'insérer une 2ème réservation, et que vous voulez créer un nouveau fichier texte, l'ancien doit être déplacé du projet.

Vous n'aurez pas besoin de plus d'explications, l'application vous guidera tout au long de son utilisation par des messages vous indiquant ce que vous devez faire.

En cas d'exécution d'une action non permise ou qui n'est pas tout à fait correcte, vous verrez s'afficher sur votre écran des messages d'erreur correspondants à l'action que vous venez d'exécuter.

L'affichage d'un message d'erreur vous renverra dans la plupart des temps vers le menu pour refaire l'action en question, car la première action ne sera pas exécutée (erreur commise en introduisant les dates ou un nom d'hôtel qui n'existe pas).

Pour le coté graphique, et par manque de temps, nous avons choisi de ne pas surcharger et de n'utiliser que quelques fonctions de la bibliothèque standard conio et non pas la sdl.

Voilà tout, nous souhaitons que notre travail vous plaise en ce qui concerne les fonctionnalités, étant donné que nous avons plus misé sur le côté pratique que sur le côté esthétique, nous espérons qu'elle vous sera facile d'utilisation. Prenez soin de respecter les petites règles des remarques pour une meilleure exécution sans bugs.