PROJET DE STRUCTURES DE DONNÉES

GESTION DES LOGEMENTS DU CROUS

GOLFIER Alexandre
TOFFIN Cyril
PERALDE François
SEVRET Yannis

GRP 3 & 4

2019-2020

SOMMAIRE

PARTIE I : Structures

I.Structures des fichiers

Fichier I : « Etudiant.don »

Ce fichier contient un nombre de ligne indéterminé triées par identifiant. Chaque ligne de celui-ci définit un seul étudiant comportant ces informations suivantes :

- Un **identifiant d'étudiant** sous la forme « ETUD – XXXXX »
- Un code booléen (1-vrai ou 0-non) qui détermine si l'étudiant est **boursier** ou non
- Un numéro (0 à 7) qui détermine **l'échelon de l'étudiant**
- Un code booléen (1-vrai ou 0-non) qui détermine si l'étudiant est **handicapé ou non**
- La **civilité** de l'étudiant (MR ou MME)
- **Nom, Prénom** de l'étudiant

Aperçu :

```
1 ETUD-11254 0 0 0 MME RODRIGUEZ LUPE
2 ETUD-11548 1 5 1 MR POLK CLYDE
3 ETUD-12547 0 0 0 MR HART STEPHEN
4 ETUD-15486 1 2 0 MME WILLIAMS STEPHANIE
5 ETUD-16478 0 0 1 MME ADAMS TRUDY
```

Fichier II : « Logement.don »

Ce fichier contient un nombre de ligne indéterminé non triées. Chaque ligne de celui-ci définit un seul logement comportant ces informations suivantes :

- Un **identifiant de logement** sous la forme « LOGE-XXXXX »
- Un **type de logement** (T2, T1, ST = Studio, CH = Chambre)
- Un code booléen (1-vrai ou 0-non) qui détermine si le logement est **disponible ou non**
- Un second code booléen (1-vrai ou 0-non) qui détermine si le logement est **adapté pour une personne à mobilités réduite**
- Un **identifiant d'étudiant** si celui-ci est habité ou « XXXXXXXXXX » si celui-ci est disponible

-La **Cité** où se situe le logement

Aperçu :

```
1 LOGE-21547 T2 1 0 XXXXXXXXXX Cite U Arc de Meyran
2 LOGE-65214 ST 0 1 ETUD-11548 Cite U Alice Chatenoud
3 LOGE-41236 CH 0 0 ETUD-65214 Cite Lucien Cornil
4 LOGE-59874 T1 0 0 ETUD-49654 Residence Canada
5 LOGE-84561 T2 0 0 ETUD-31254 Cite U Galinat
```

Fichier III : « Demande.don »

Ce fichier contient un nombre de ligne indéterminé triées par échelon. Chaque ligne de celui-ci définit une seule demande comportant ces informations suivantes :

- Un **identifiant de demande** sous la forme « DEMA-XXXXX »
- Un **type de logement** (T2, T1, ST = Studio, CH = Chambre)
- **L'echelon** de l'étudiant demandeur (0 à 7)
- Un code booléen (1-vrai ou 0-non) qui détermine si l'étudiant demandeur possède un **handicape ou non**
- **L'identifiant** de l'étudiant demandeur
- La **Cité** où l'étudiant souhaite résider

Aperçu :

```
1 DEMA-25139 T2 7 0 ETUD-58012 Cite U Arc de Meyran
2 DEMA-09012 ST 7 1 ETUD-26987 Cite Claude Delorme
3 DEMA-69530 T1 6 0 ETUD-25478 Residence Canada
4 DEMA-76621 T2 6 0 ETUD-37542 Cite Lucien Cornil
```

Fichier IV : « Cite.don »

Ce fichier contient un nombre de ligne déterminé et précisé dès la première ligne de celui-ci. Chaque ligne est essentiellement constituée d'un **Nom de cité**.

Aperçu :

```
1 9
2 Cite U Arc de Meyran
3 Cite U Alice Chatenoud
4 Cite U Claude Delorme
5 Cite U Galinat
```

II. Définition des structure

Pour pouvoir lire et charger les fichiers, plusieurs structures ont été définies ;

Structure I : Etudiant

Cette structure contient :

- une **chaîne de caractères** pour l'identifiant étudiant
- Trois **entier** représentant la bourse, l'échelon ainsi que l'handicape de l'étudiant
- une **chaîne de caractères** pour la civilité de celui-ci
- une **chaîne de caractères** contenant le nom et le prénom de l'étudiant

Déclaration :

```
typedef struct
{
        char IdEtudiant[11] ;
        int Bourse, Echelon, Handicap ;
        char Civilite[4], Nom[40] ;
}Etudiant ;
```

Structure II : Logement

Cette structure contient :

- Une **chaîne de caractères** pour l'identifiant logement
- Une **chaîne de caractères** pour le type de logement
- Deux **entier** représentant la disponibilité et l'adaptation pour les handicapés
- Une **chaîne de caractères** pour l'identifiant de l'étudiant
- Une **chaîne de caractères** pour le nom de la cite

Déclaration :

```
typedef struct
{
        char IdLogement[11], TypeLog[2] ;
        int Disponible, AdaptHandicap ;
        char IdEtudiant[11],Cite[30] ;
}Logement ;
```

Structure III : Demande

Cette structure contient :

- Une **chaîne de caractères** pour l'identifiant de demande
- Une **chaîne de caractères** pour le type de logement demandé
- Deux **entier** représentant l'échelon de l'étudiant demandeur et son handicape
- Une **chaîne de caractères** pour l'identifiant de l'étudiant demandeur
- Une **chaîne de caractères** pour la cité demandée

Déclaration :

```
typedef struct
{
        char IdDemande[11], TypeLog[2] ;
        int EchelonEtudDem, Handicap ;
        char IdEtudiant[11] , Cite[30] ;
}Demande ;
```

Structure IV : Cite

Cette Structure contient essentiellement une **chaîne de caractères** pour un Nom de Cite.

Declaration :

```
typedef struct
{
        char Nom[30]
}Cite ;
```

Dès lors, pour stocker les informations de chaque fichier, nous disposons de 4 tableaux ayant été déclaré de type Etudiant, Logement, Demande ou Cite. A la différence des autres, l'allocation d'espace du tableau de type Cite se fait directement dans la fonction chargement et non dans la fonction test comme les autres.

PARTIE II : Fonctionnalités

Fonctionnalité I : Chargement des fichiers

Etat : Marche

Pour cette fonctionnalité, 4 fonctions chargement ont été mises en place et nous pouvons retrouver deux types de chargement parmi ces fonctions :

- **Chargement à allocation dynamique**

Les tableaux, Étudiant, Logement et Demande sont chargés dynamiquement. En effet, une allocation (**malloc)** de 5 espaces est réalisée pour ces 3 tableaux dans la fonction test. Au moment de rentrer dans une fonction Chargement, ces tableaux possèdent une taille logique de 0 et une taille physique de 5.

Une fois dans une fonction Chargement, si la taille logique est égale à la taille physique au moment d'insérer une nouvelle valeur dans le tableau, une réallocation d'espace (realloc) est réalisée copiant le tableau et rajoutant 5 espaces supplémentaires à celui-ci.

Lorsque le chargement est terminé, le tableau rempli est retourné dans la fonction test.

- **Chargement à allocation d'espace fixe**

Le tableau Cite est le seul tableau où nous connaissons le nombre d'élément qui le compose et nous savons même que ce nombre est précisé à la première ligne de celui-ci. Une fois dans la fonction chargement celle-ci va lire la première ligne du fichier et faire une allocation (**malloc**) d'espace égale au nombre lu dans le fichier.

Ce même nombre va créer une boucle et faire en sorte que la fonction lise le bon nombre de ligne et ainsi éviter que la taille logique ne dépasse la taille physique du tableau et créer une erreur.

Fonctionnalité II : Affichage des logements disponibles par cité

Etat : Marche

Cette fonctionnalité permet d'avoir un affichage trié, par cité, des logements disponibles. Pour se faire, la fonction appelée va pour chaque cité, parcourir tout les logements présent dans le tableau . Et pour chaque logement, vérifier si celui-ci est disponible et si la cité du logement est identique à la cité traitée pour ensuite l'afficher dans le cas où les conditions sont respectées.

Fonctionnalité III : affichage de la liste des logements occupés en mentionnant l'identité de l'étudiant qui l'occupe

Etat : Marche

Cette fonctionnalité permet d'avoir un affichage des logements occupés en mentionnant le Nom de l'étudiant qui l'occupe. Pour se faire, la fonction appelée va pour chaque logement occupé, parcourir tout les étudiants présent dans le tableau. Et pour chaque étudiant, vérifier si l'identifiant de l'étudiant occupant le logement est identique que celui de l'étudiant traité pour ensuite l'afficher dans le cas où la condition est respectée.

Fonctionnalité IV : affichage des demandes de logement en attente

Etat : Marche

Cette fonctionnalité permet d'afficher toutes les demandes ainsi que toutes les informations liées à celle-ci présentes dans le tableau Demande.

Fonctionnalité V : Traitement des demandes en attentes

Etat : Marche

Cette fonctionnalité permet de traiter une seule demande avant de retourner dans la fonction test. Pour se faire, la fonction appelée va commencer par traiter la première demande en attente. Elle va ensuite pour chaque logement vérifier si celui-ci est disponible et si la cité ainsi que le type de logement sont les mêmes que la demande en traitement.

Dans le cas, où, toutes ces conditions sont respectées, le logement va être attribué à l'étudiant demandeur et sa demande va être supprimé du tableau. Puis les tableaux mis à jour vont être retournés dans la fonction test.

Dans le cas contraire, l'utilisateur va avoir le choix de lancer une seconde fonction nous proposant quelques conseils. En effet cette fonction va afficher les logements du même type souhaité ne se situant pas dans la cité demandée et les logement se situant dans la bonne cité mais n'étant pas du même type souhaité.

Si l'utilisateur refuse d'avoir un conseil, la fonction va traiter la prochaine demande en attente et recommencer le même processus.

Sinon, l'utilisateur va avoir le choix entre plusieurs logements respectant les conditions ci-dessus.

A ce moment, il doit sélectionner un numéro attribué à un logement pour l'affecter à la demande et la supprimer. Mais malgré les différents conseil, l'utilisateur peut encore abandonner le traitement de la demande et ainsi commencer à traiter la demande en attente suivante et recommencer le processus.

Lors de l'attribution d'un logement à un étudiant demandeur, la fonction supprime toutes les demandes comportant le même identifiant étudiant pour éviter que celui-ci soit dans plusieurs logements.

Si toutes les demandes sont traitées, un message s'affichera.

Fonctionnalite VI : Saisie d'une nouvelle demande

Etat : Marche

Cette fonctionnalité permet de créer une nouvelle demande et l'insérer dans le tableaux trié. Plusieurs vérification de saisie on été mis en place pour cette fonction, il est donc impossible d'avoir un identifiant de demande en doublon car celui-ci est unique à une demande. Lors de la saisie de l'identifiant étudiant, celui-ci peut déjà exister, l'utilisateur à le choix d'arrêter la fonction ou de continuer la création de demande.

Dans le cas où l'étudiant existe et que l'étudiant réside déjà dans un logement. La fonction s'arrête et retourne dans la fonction test. Si il existe mais qu'il ne réside dans aucun logement, les informations nécessaire à la création demande vont être envoyées.

Si l'étudiant n'existe pas, une création va être lancée, l'insérant ensuite dans le tableau étudiant trié et compléter la demande avec les informations nécessaires du nouvel étudiant.

Une fois la création terminée, la demande va être insérée dans le tableau trié par échelon d'étudiant avant de retourner le tableau mis à jour dans la fonction test.

Fonctionnalité VII : Annulation d'une demande

Etat : Marche

Cette fonctionnalité permet d'annuler une demande non traitée. Pour se faire la liste des demandes va être affichée, l'utilisateur devra seulement saisir l'identifiant de la demande qu'il souhaite supprimer. Si l'identifiant n'existe pas, un message d'erreur s'affiche et l'utilisateur doit ressaisir un bon identifiant. Si l'identifiant existe, une suppression est faite dans le tableau et la fonction retourne le tableau mis à jour.

Remarque :

Si la différence entre la taille physique et la taille logique dans le tableau Demande est égale à cinq, une réallocation d'espace (realloc) va être réalisée pour ainsi éviter qu'il y trop d'espace vide.

Fonctionnalité VIII : Libération de logement

Etat : Marche

Cette fonctionnalité permet de libérer un logement occupé. Pour se faire, l'utilisateur devra rentrer l'identifiant du logement qu'il souhaite libérer. Si l'identifiant n'existe pas, il devra ressaisir un identifiant. Si l'identifiant existe et que le logement est déjà libre, la fonction s'arrête en retournant le tableau de logement.

Remarque :

Si le logement est occupé, la libération va être réalisée et une seconde fonction va être lancée dans le but de trouver une demande de logement susceptible de correspondre au logement libéré. Si une demande compatible est trouvée, l'utilisateur va avoir le choix d'attribuer ou non le logement libéré à l'étudiant en supprimant sa demande. Sinon, la fonction retourne le tableau mis à jour avec un logement libre en plus.

Fonctionnalité IX : Sauvegarde

Etat : Marche

Cette fonctionnalité, une fois lancée, permet d'enregistrer toutes les modifications réalisées durant l'ouverture du programme dans les fichiers vus avant.