# **Projet de structures de données : Gestion des logements du CROUS**

## Définition des structures :

typedef enum{faux,vrai}Booleen;

typedef struct

{

int IdLogement;

char Nom[30];

int Type;

Booleen Dispo;

Booleen Adapte;

int IdEtudiant;

} Logement;

typedef struct

{

int IdEtudiant;

char Civilite[4];

char Nom[20];

char Prenom[20];

Booleen Boursier;

int Echelon;

Booleen Handicap;

} Etudiant;

typedef struct

{

int IdDemande;

int IdEtudiant;

int Echelon;

char Nom[20];

int Type;

} Demande;

typedef struct maillonDemande

{

Demande demande;

struct maillonDemande \*suiv;

}MaillonDemande,\*ListeDemande;

# Fichiers textes :

**Fichier Logements :**

6

2 3 1 1 5

Cezeaux

3 1 1 0 6

Le Clos

4 0 1 0 2

Crous

6 1 1 0 3

Le Clos

5 1 0 1

Crous

1 3 1 1 1

Cezeaux

**Fichier Etudiant :**

1 Mr

Da Silva

Thomas

0 0 0

2 Mme

Bollecker

Collincia

1 5 1

3 Mr

Chaumuzeau

Oscar

1 1 0

4 Mr

Periquoi

Yoann

1 3 1

5 Mme

Dupont

Adele

0 0 0

6 Mme

Martin

Elise

1 7 1

**Fichier Demandes :**

4 5 0 0

Crous

2 6 7 2

Le Clos

3 2 5 1

Crous

6 3 1 2

Le Clos

1 4 3 0

Cezeaux

5 1 0 3

Cezeaux

# **Représentation dans la mémoire centrale :**

## 

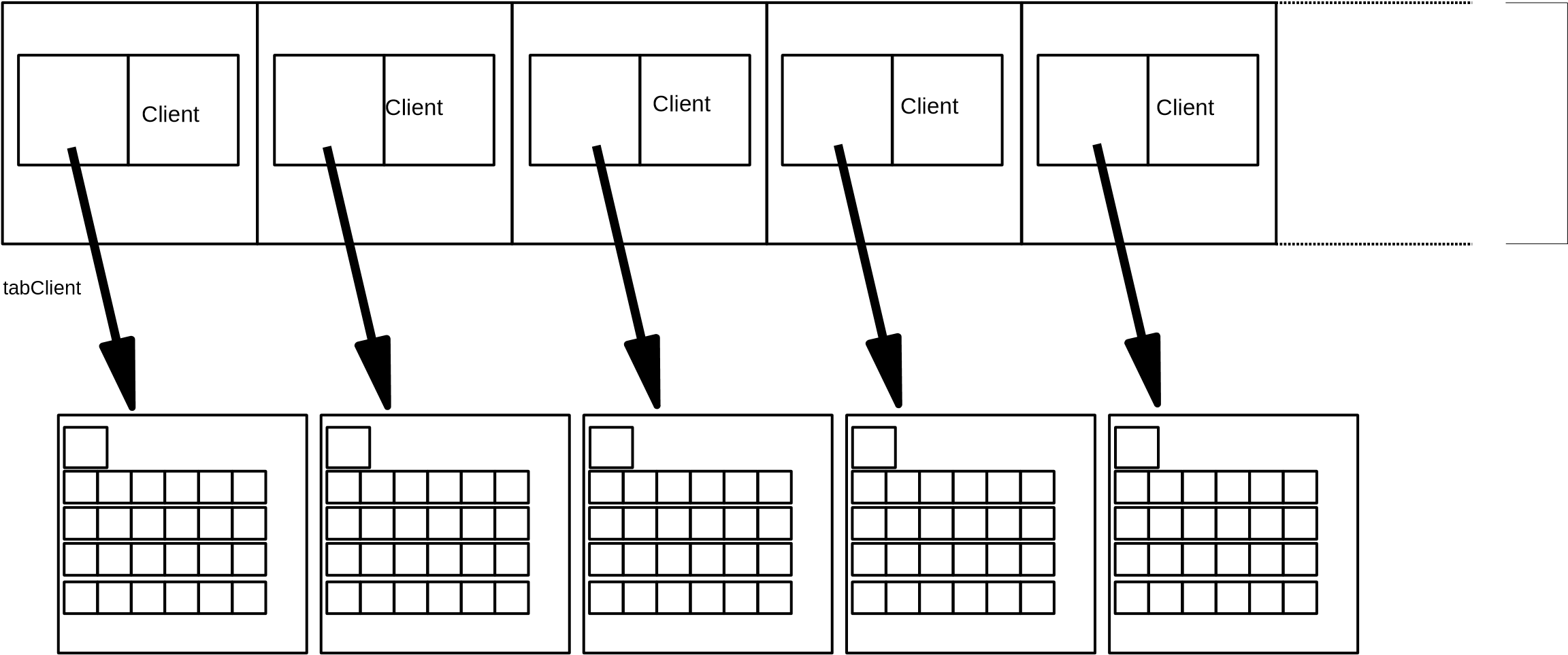
## Liste chaînée linéaire de structure demande :

## 

## 



### Tableau dynamique de pointeurs sur des structures étudiants :





## Arborescence :

**Menu**

**├── Chargement**

│ **├─ Logement**

│ **├─ Etudiant**

│ **├─ Demandes**

│

│

│

├**── Afficher**

│ **├─** **Étudiants**

│ **├─ Logements**

│ **├─ Logements triés par citées**

│ **├─ Logements occupés**

│ **├─ Demandes**

│

## **├──** **Fonctions**

## │ **├─** **traitement des demandes**

## │ **├─ Créer une demande de logement**

## │ **├─ Création d’un étudiant**

## │ **├─ Annulation d’une demande**

│ **├─ Libération de Logement**

## **├──** **Sauvegarde**

**├── Logement**

**├── Etudiant**

**├── Demandes**

# Fonctions :

## Chargement des fichiers texte :

Etudiants :

Charge le fichier texte Etudiant en tant que tableau dynamique de pointeur sur des structures.

Int IdEtudiant;

char Civilite[4];

char Nom[20];

char Prenom[20];

Booleen Boursier;

int Echelon;

Booleen Handicap;

Etudiant \*\*chargerFetudiant (int \*nb,int \*max)

* Appelle la fonction qui lit un étudiant : Etudiant lireEtud(FILE \*fe)
* Retourne l’adresse du tableau de pointeur.

Logements :

Charge le fichier texte Logement en tant que tableau statique de structure, chaque structure a la forme :

int IdLogement;

char Nom[30];

int Type;

Booleen Dispo;

Booleen Adapte;

int IdEtudiant;

* Retourne le nombre de logements chargés dans le tableau.

Demandes :

Charge le fichier binaire demandeEnAttente en tant que liste chaîné linéaire :

Chaque maillon est structuré comme ceci :

int IdDemande;

int IdEtudiant;

int Echelon;

char Nom[20];

int Type;

Chaque maillon est défini de la façon suivante :

Typedef struct maillonDemande

{

Demande demande;

struct maillonDemande \*suiv;

}MaillonDemande,\*ListeDemande;

ListeDemande chargerFdemande(int \*nb)

-> lis le fichier.

-> appelle listeNouv() pour créer une liste.

-> appelle lireDem(flot) pour lire une demande.

-> tant que le programme lis dans le fichier : insertion(l,d)

## Affichage :

1. Etudiant :

void afficherEtudiants(Etudiant \*\*tetud,int nb)

Affiche les étudiants, pour cela appelle la fonction :

void afficher1Etudiant(Etudiant tetud)

Cette fonction est appelée tant que la boucle n’a pas parcouru tous les étudiants soit nb.

1. Logements :

void afficherLogement(Logement \*tlog,int nb)

Affiche les logements, pour cela appelle la fonction :

void afficher1Logement(Logement l)

Cette fonction est appelée tant que nb n’est pas atteint.

1. Logements occupés :

void affichageLogementOccupe(Logement \*tlog,Etudiant \*\*tetud,int nbLog,int nbEtud)

Recherche les logements occupés avec une recherche dichotomique:

int rechDico(Etudiant \*\*tetud, int n,int val)

Si le logement est occupé les fonctions suivantes sont exécutées pour ce dernier :

afficher1Logement(tlog[i]);afficher1Etudiant(\*tetud[rech]);

1. Logement triés

Trie la liste des logements par cité :

void trifusion (Logement tlog[],int n)

Pour cela appelle la fonction void fusion (Logement r[], int n, Logement s[], int m, Logement tlog[]) et la fonction void copie (Logement tlog[], int i,int j, Logement r[]

1. Demandes :

void afficherDemandes(ListeDemande l,int nb)

Apelle la fonction suivante en boucle tant que nb n’est pas atteint :

void afficher1Demande(Demande d)

## Traitement des demandes en attentes :

Traite les demandes en cours, rends iindiponible les logements attribués et supprime les demandes:

void traitementDemandes(Etudiant \*\*tetud,Logement \*tlog,ListeDemande l, int nbEtud, int nbLog, int nbDem)

## Nouvelle demande de logement :

Permet de créer une nouvelle demande de logement avec la création d’un nouvel etudiant. La fonction vérifie aussi si le nouvel id de la demande est déjà existant.

Void creationDemande(Logement tlog[], Etudiant \*\*tetud, ListeDemande l, int \*nbDem, int \*nbEtud, int \*nbLog, int \*maxEtud)

Appel la fonction rechercheDemande (ListeDemande l, int val) pour vérifier si la demande qui veut être créer par l’utilisateur n’est pas déjà existante.

## Annulation d’une demande non traitée :

Permet de supprimer une demande si elle n’a pas encore été traitée :

ListeDemande annulationDemande(int \*nb,ListeDemande l)

Appelle : -> ListeDemande suppression (ListeDemande l, int x,int \*nb)

## Libération d’un logement lors d’un départ :

Logement \*liberationlogement(Etudiant \*\*tetud,Logement \*tlog,ListeDemande l, int nbEtud, int \*nbLog, int \*nbDem)

Demande quel logement n’est plus occupé et change son statut d’occupation.

Pour cela recherche le logement en fonction de son id logement avec la fonction:

rechercherLogement(Logement tlog[],int nb, int val,int \*trouve)

## Sauvegarde des données des trois fichiers :

A) Sauvegarde du fichier texte Etudiant:

sauveTabEtudiant(tetud,nb);

B) Sauvegarde du fichier texte Logement :

sauveTablogement(tlog,nb);

C)Sauvegarde du fichier texte Demande :

sauveDem(l,nb);

D)Sauvegarde du fichier binaire Demande :

sauveTResultbin(l,nb);