# IMG_256Kouyou moughandegha ange

# Encadree par mr Kamal Omari

# Rapport d’Analyse du Système de Gestion Hospitalière

## Table des matières

1. [Introduction](#introduction)
2. [Architecture du Système](#architecture-du-système)
   * [Structures de Données](#structures-de-données)
   * [Organisation des Données](#organisation-des-données)
3. [Analyse des Fonctionnalités](#analyse-des-fonctionnalités)
   * [Gestion des Hôpitaux](#gestion-des-hôpitaux)
   * [Gestion des Patients](#gestion-des-patients)
   * [Gestion des Maladies](#gestion-des-maladies)
4. [Persistance des Données](#persistance-des-données)
5. [Interface Utilisateur](#interface-utilisateur)
6. [Analyse Critique et Améliorations Possibles](#Xc92e939fc44303184b7a18daf8e5383d749ad14)
7. [Conclusion](#conclusion)

## Introduction

Ce rapport présente une analyse détaillée d’un système de gestion hospitalière développé en langage C. Le programme permet de gérer un réseau d’hôpitaux, leurs patients et les maladies traitées. Il utilise différentes structures de données pour organiser ces informations et offre diverses fonctionnalités pour manipuler ces données.

Le système est conçu autour de trois entités principales :

* Les hôpitaux (représentés par une liste doublement chaînée)
* Les patients (représentés par une file)
* Les maladies (représentées par une pile)

Cette application permet de créer, afficher, modifier et supprimer ces différentes entités, ainsi que d’effectuer des recherches et des tris sur les données.

## Architecture du Système

### Structures de Données

Le système utilise trois structures de données principales pour représenter les entités :

#### Structure Maladie (Pile)

typedef struct Maladie{  
 char nom\_maladie[100];  
 char symptome[100];  
 int dangerosite;  
 char traitements[100];  
 struct Maladie \*suivant;  
}Maladie;  
typedef Maladie\* liste\_maladie;

Cette structure représente une maladie avec ses caractéristiques (nom, symptômes, niveau de dangerosité, traitements) et est implémentée sous forme de pile.

#### Structure Patient (File)

typedef struct patient{  
 char nom\_patient[100];  
 char ID[100];  
 int age;  
 Maladie data;  
 char status[100];  
 char chambre[100];  
 char nom\_hopital[100];  
 struct patient \*suivant;  
}patient;  
typedef patient \*liste\_patient;

Cette structure représente un patient avec ses informations personnelles (nom, ID, âge, statut, chambre) et les informations sur sa maladie. Elle est implémentée sous forme de file.

#### Structure Hôpital (Liste doublement chaînée)

typedef struct hopital{  
 char nom\_hopital[100];  
 char adresse[100];  
 char numero\_telephone[100];  
 char email[100];  
 char nomdepartements[100];  
 int nombretotaldelits;  
 char horaireouverture[3];  
 liste\_patient headpatienthopital;  
 liste\_maladie headMaladiehopital;  
 struct hopital \*suivant;  
 struct hopital \*precedent;  
}hopital;  
typedef hopital\* liste;

Cette structure représente un hôpital avec ses informations (nom, adresse, numéro de téléphone, email, départements, nombre de lits, horaires d’ouverture) et contient des pointeurs vers la liste des patients et la liste des maladies traitées dans cet hôpital.

### Organisation des Données

Le système organise les données de manière hiérarchique :

* Une liste d’hôpitaux est maintenue (liste doublement chaînée)
* Chaque hôpital contient une liste de patients (implémentée comme une file)
* Chaque hôpital contient également une liste de maladies traitées (implémentée comme une pile)

Cette organisation permet d’accéder facilement aux patients et aux maladies pour un hôpital spécifique.

## Analyse des Fonctionnalités

### Gestion des Hôpitaux

#### Création d’un Hôpital

liste creer\_hopital()

Cette fonction permet de créer un nouvel hôpital en demandant à l’utilisateur de saisir les informations nécessaires (nom, adresse, numéro de téléphone, départements, nombre de lits, horaires d’ouverture). Elle crée également un dossier pour l’hôpital dans le système de fichiers et y stocke les informations de l’hôpital dans un fichier texte.

Points clés de la fonction :

* Allocation dynamique d’une structure d’hôpital
* Saisie des informations de l’hôpital
* Création d’un dossier pour l’hôpital
* Sauvegarde des informations dans un fichier texte
* Initialisation des listes de patients et de maladies à NULL

#### Ajout d’un Hôpital au Début de la Liste

liste Ajout\_debut\_hopitaux(liste head)

Cette fonction permet d’ajouter un nouvel hôpital au début de la liste des hôpitaux. Elle appelle la fonction creer\_hopital() pour créer un nouvel hôpital, puis l’ajoute au début de la liste.

#### Ajout d’un Hôpital à la Fin de la Liste

liste Ajout\_Fin\_hopitaux(liste head)

Cette fonction permet d’ajouter un nouvel hôpital à la fin de la liste des hôpitaux. Elle appelle la fonction creer\_hopital() pour créer un nouvel hôpital, puis l’ajoute à la fin de la liste.

#### Suppression d’un Hôpital au Début de la Liste

liste Supprimerdebuthopital(liste head)

Cette fonction permet de supprimer l’hôpital situé au début de la liste des hôpitaux. Elle gère correctement les cas où la liste est vide ou ne contient qu’un seul élément.

#### Suppression d’un Hôpital à la Fin de la Liste

liste SupprimerFin(liste head)

Cette fonction permet de supprimer l’hôpital situé à la fin de la liste des hôpitaux. Elle vérifie si la liste est vide avant de procéder à la suppression.

#### Affichage des Hôpitaux

void afficherhopital(liste head)

Cette fonction affiche les informations de tous les hôpitaux de la liste, y compris les informations de leurs patients respectifs.

### Gestion des Patients

#### Ajout d’un Patient à un Hôpital

liste AjouterpatientFin(liste head, const char\* nom\_hopital)

Cette fonction permet d’ajouter un nouveau patient à un hôpital spécifié par son nom. Elle demande à l’utilisateur de saisir les informations du patient et les ajoute à la liste des patients de l’hôpital correspondant. Si le patient est malade, la fonction crée également une nouvelle maladie et l’ajoute à la liste des maladies de l’hôpital.

Points clés de la fonction :

* Recherche de l’hôpital correspondant dans la liste
* Allocation dynamique d’une structure de patient
* Saisie des informations du patient
* Gestion différente selon que le patient est malade ou visiteur
* Ajout du patient à la fin de la liste des patients de l’hôpital

#### Suppression d’un Patient

liste SupprimpatientFin(liste head)

Cette fonction permet de supprimer le premier patient de la liste des patients d’un hôpital. Elle gère correctement les cas où la liste de patients est vide ou ne contient qu’un seul élément.

#### Création de Fichiers Patients

void creer\_fichier\_patient(const char \*chemin\_dossier\_hopital, const patient \*patient\_info)

Cette fonction crée un fichier texte pour un patient dans le dossier de l’hôpital correspondant. Le fichier contient toutes les informations du patient, y compris les informations sur sa maladie s’il est malade.

#### Ajout de Fichiers Patients pour Tous les Hôpitaux

void ajouter\_fichiers\_patients\_hopitaux(liste head)

Cette fonction parcourt tous les hôpitaux et crée des fichiers pour tous les patients de chaque hôpital. Elle appelle la fonction creer\_fichier\_patient() pour chaque patient.

### Gestion des Maladies

#### Suppression de Toutes les Maladies

liste Supprim\_maladieTOUTE(liste head)

Cette fonction supprime toutes les maladies de tous les hôpitaux. Elle parcourt la liste des hôpitaux et, pour chaque hôpital, supprime toutes les maladies de sa liste de maladies.

#### Tri des Maladies par Ordre Croissant

liste trier\_ordrecroissant\_maladie(liste head)

Cette fonction trie les maladies de chaque hôpital par ordre croissant de dangerosité. Elle utilise l’algorithme de tri à bulles pour effectuer le tri.

#### Tri des Maladies par Ordre Décroissant

liste trier\_ordredecroissant\_maladie(liste head)

Cette fonction trie les maladies de chaque hôpital par ordre décroissant de dangerosité. Elle utilise également l’algorithme de tri à bulles.

#### Recherche d’une Maladie

void Rechercher\_Maladie(liste head, const char \* nom\_maladie)

Cette fonction recherche une maladie spécifique dans tous les hôpitaux et affiche la liste des hôpitaux qui traitent cette maladie.

#### Affichage des Maladies par Hôpital

void affichermaladieparhopitaux(liste head)

Cette fonction affiche toutes les maladies traitées dans chaque hôpital.

## Persistance des Données

Le système assure la persistance des données grâce à plusieurs fichiers :

* Un fichier nom\_hopitaux.txt qui contient la liste des noms de tous les hôpitaux
* Un fichier chemin.txt qui contient les chemins vers les dossiers des hôpitaux
* Un fichier chemindossierpatient.txt qui contient les chemins vers les dossiers des patients
* Un dossier par hôpital, contenant un fichier avec les informations de l’hôpital
* Un fichier par patient dans le dossier de l’hôpital correspondant

Ces fichiers permettent de stocker les informations sur les hôpitaux, les patients et les maladies, même après la fermeture du programme.

## Interface Utilisateur

L’interface utilisateur est basée sur des menus textuels qui permettent à l’utilisateur de naviguer entre les différentes fonctionnalités du système :

int menu()

Le menu principal qui permet de choisir entre la gestion des hôpitaux, des maladies ou des patients.

int menuhopital()

Le menu de gestion des hôpitaux qui permet d’ajouter, d’afficher ou de supprimer des hôpitaux.

int menuMaladie()

Le menu de gestion des maladies qui permet de supprimer, d’afficher, de trier ou de rechercher des maladies.

int menuPatient()

Le menu de gestion des patients qui permet d’ajouter, d’afficher ou de supprimer des patients.

L’interface utilise également des codes de couleur pour améliorer la lisibilité des informations affichées.

## Analyse Critique et Améliorations Possibles

Le système de gestion hospitalière présente plusieurs points forts :

* Organisation hiérarchique claire des données
* Utilisation de structures de données adaptées aux besoins (liste doublement chaînée, file, pile)
* Persistance des données grâce aux fichiers
* Interface utilisateur simple et intuitive

Cependant, plusieurs améliorations pourraient être apportées :

* Validation des entrées utilisateur pour éviter les erreurs
* Gestion des erreurs plus robuste
* Séparation plus claire entre la logique métier et l’interface utilisateur
* Possibilité de modifier les informations des hôpitaux, des patients et des maladies
* Ajout de fonctionnalités de recherche plus avancées
* Sécurisation des données sensibles
* Utilisation de chemins relatifs plutôt que de chemins absolus pour les fichiers

## Conclusion

Le système de gestion hospitalière présenté dans ce rapport offre une base solide pour la gestion d’un réseau d’hôpitaux, de leurs patients et des maladies traitées. Il utilise des structures de données appropriées et des fonctionnalités de base pour manipuler ces données.

Bien que le système puisse être amélioré sur plusieurs aspects, il fournit déjà une solution fonctionnelle pour la gestion hospitalière à petite échelle. Les améliorations suggérées pourraient être implémentées dans une version future du système pour le rendre plus robuste et plus facile à utiliser.