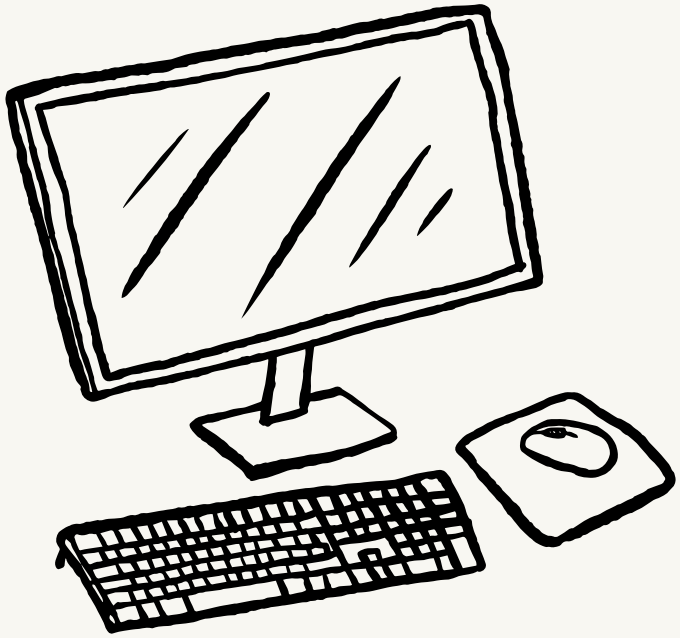
Rapport du projet SDD

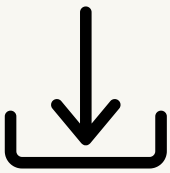
Gestion de contacts

ELOUAQQAD MOHAMMED
OUSSAMA ELIDRYSY
HAJJAJI AHMED YASSINE
EZ-ZAOUI NASR-EDDINE



professeur Khadija
OUAZZANI TOUHAMI

SOMMAIRE _



1. INTRODUCTION ET REMERCIEMENTS _

1.1 Contexte du projet

Ce projet consiste en la conception et le développement d'un gestionnaire de contacts en langage C. L'objectif principal est de créer une application capable de gérer efficacement une base de données de contacts personnels, permettant l'ajout, la modification, la suppression, la recherche et la sauvegarde des informations de contact.

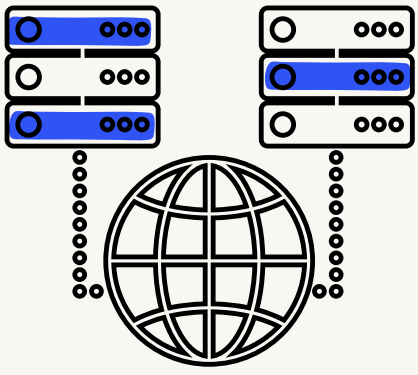
1.2 Objectifs du projet

- Développer une interface utilisateur intuitive en mode console
- Implémenter une structure de données efficace pour le stockage des contacts
- Permettre la gestion complète des contacts (CRUD : Create, Read, Update, Delete)
- Offrir des fonctionnalités de recherche avancées
- Assurer la persistance des données via la sauvegarde en fichier

1.3 Cahier des charges

Le système doit permettre de :

- Ajouter de nouveaux contacts avec nom, téléphone et email
- Supprimer des contacts existants
- Modifier les informations d'un contact
- Afficher tous les contacts dans l'ordre alphabétique
- Rechercher des contacts par différents critères
- Sauvegarder et charger les données depuis un fichier



5. DESCRIPTION DU TRAVAIL DE NOTRE ÉQUIPE SUR LE PROJET _

Phase de conception théorique

Avant de commencer la programmation, nous avons pris le temps de faire une étude théorique approfondie du projet. Cette phase nous a permis de bien comprendre les besoins et les fonctionnalités à implémenter. En équipe, nous avons discuté et défini :

- Les structures de données nécessaires (notamment une structure Compte contenant les informations comme le nom, prénom, identifiant, solde, etc.).
- Les fonctions principales à développer, telles que : ajouter un compte, modifier un compte, rechercher un compte (par la première lettre, les deux premières lettres, ou le nom complet), afficher les comptes, sauvegarder et charger les données depuis un fichier.
- La méthode d'implémentation, en optant pour une gestion via des listes chaînées afin de permettre une manipulation dynamique des comptes.

Cette préparation nous a permis d'aborder le codage avec une vision claire et organisée.

Phase de développement individuel en équipe

Une fois la conception achevée, nous avons réparti les tâches entre les membres de l'équipe afin de travailler en parallèle sur différentes fonctionnalités du projet. Chaque membre s'est vu attribuer une mission précise :

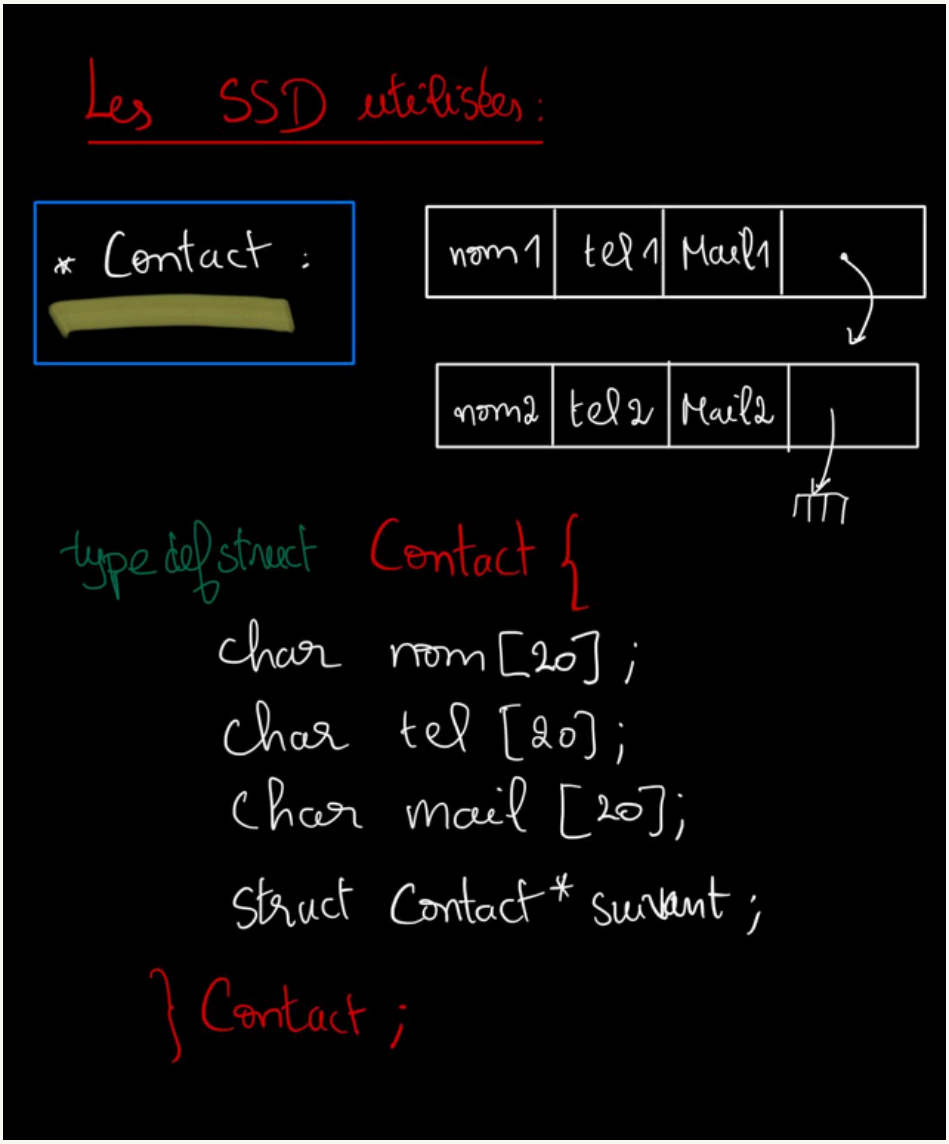
- Un membre s'est chargé de l'implémentation de l'ajout et de la suppression des comptes.
- Un autre a développé les fonctions de recherche avec les différents critères (lettres et nom complet).
- Un autre a travaillé sur la sauvegarde et le chargement des comptes depuis un fichier.
- Et enfin, un autre a assuré la création de l'interface console et l'interaction avec l'utilisateur.

Cette méthode de travail nous a permis de gagner du temps tout en responsabilisant chaque membre de l'équipe.

Phase d’intégration, de test et d’amélioration

Une fois chaque partie du code terminée, nous avons procédé à une intégration progressive des différents modules. Cette étape a nécessité :
La fusion du code en un seul fichier principal cohérent.
L’exécution de nombreux tests pour identifier les erreurs (bugs) et les incompatibilités.
Des corrections collaboratives afin de résoudre les problèmes détectés.
Des améliorations sur la clarté du code, l’ergonomie de l’interface console, et l’optimisation des fonctions.
Ce travail d’intégration et de validation a été essentiel pour garantir la stabilité et l’efficacité de notre programme.

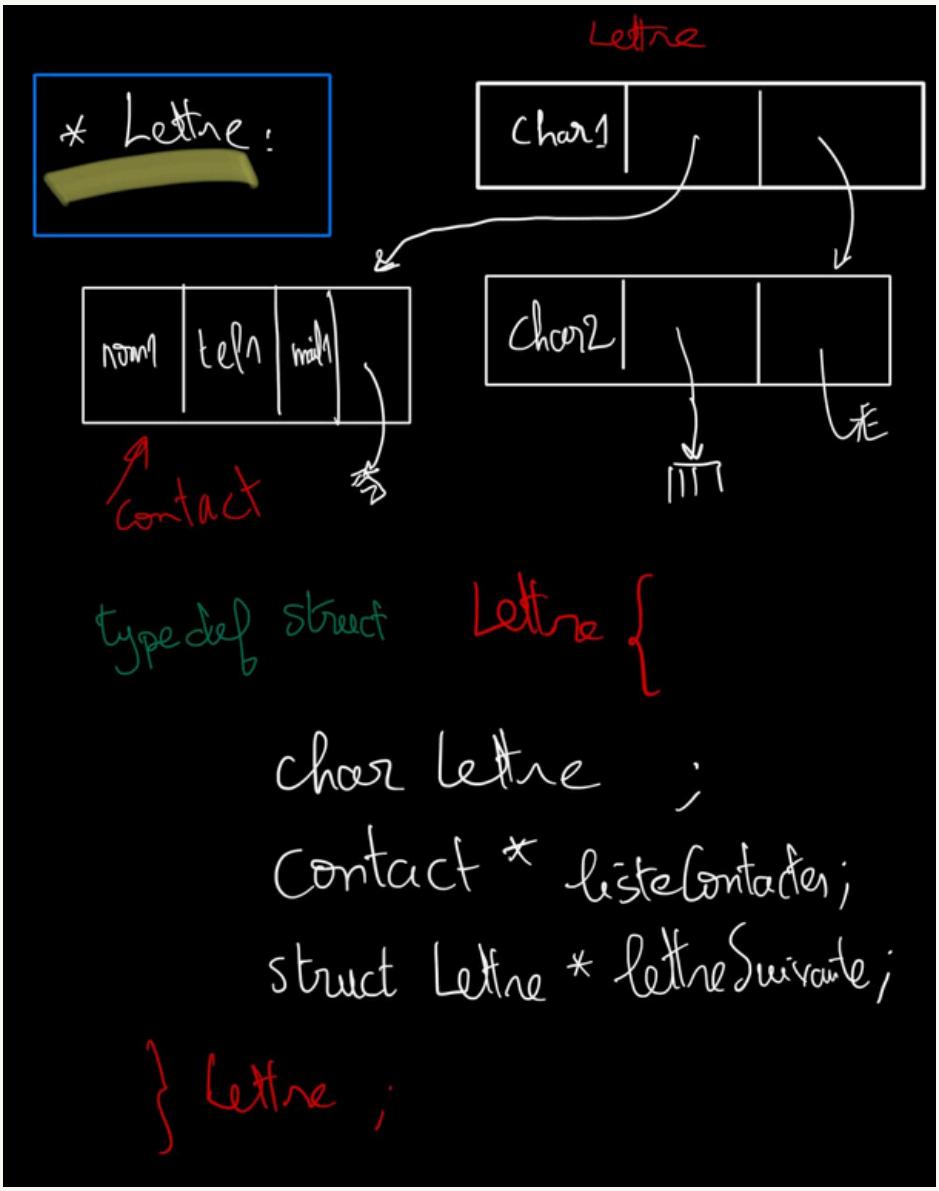
2. SDDS UTULISEES _



✅ **Structure Contact :**

La structure Contact permet de stocker les informations d'un contact. Elle contient :

- **char nom[20] : le nom du contact,**
- **char tel[20] : le numéro de téléphone,**
- **char mail[20] : l'adresse e-mail,**
- **struct Contact* suivant : un pointeur vers le contact suivant, ce qui permet de créer une liste chaînée de contacts.**



✅ **Structure Lettre :**

La structure Lettre permet de regrouper les contacts par la première lettre de leur nom. Elle contient :

- **char lettre : la lettre associée à ce groupe de contacts (exemple : 'A', 'B', etc.),**
- **Contact* listeContact : un pointeur vers la liste des contacts dont le nom commence par cette lettre,**
- **struct Lettre* lettreSuivante : un pointeur vers la lettre suivante, ce qui permet de créer une liste chaînée de lettres.**

Ces deux structures sont utilisées ensemble pour organiser un répertoire de contacts par ordre alphabétique.

3. ALGORITHMES DES FONCTIONS PRINCIPALES _

1 ajouterContact()

un ajout de contact par nom numero et email

```
// Fonction pour ajouter un contact (création de la lettre si besoin)
void ajouterContact(Lettre** listeLettres, const char* nom, const char* tel, const char* mail) {
    Si (nom == NULL || nom[0] == '\0')alors
        Ecrire("Nom invalide.\n");
        return;
    Finsi
    char premiereLettre = toupper(nom[0]);
    Lettre *precLettre = NULL, *courantLettre = *listeLettres;

    // Recherche de la lettre ou de la position d'insertion
    Tantque (courantLettre && courantLettre->lettre < premiereLettre)faire
        precLettre = courantLettre;
        courantLettre = courantLettre->lettreSuivante;
    FTQ

    // Cas 1 : La lettre n'existe pas, il faut la créer et l'insérer
    Si (!courantLettre || courantLettre->lettre != premiereLettre) alors
        Lettre* nouvelleLettre = malloc(sizeof(Lettre));
    Sinon
        Si (nouvelleLettre == NULL)alors
            printf("Erreur d'allocation memoire.\n");
            exit(1);
        Finsi
    Finsi
        nouvelleLettre->lettre = premiereLettre;
        nouvelleLettre->listeContacts = NULL;
        nouvelleLettre->lettreSuivante = courantLettre;

        Si(precLettre == NULL)alors
            // Insertion en tête
            *listeLettres = nouvelleLettre;
        Sinon
            precLettre->lettreSuivante = nouvelleLettre;
        Finsi
        courantLettre = nouvelleLettre;
    }
```

```
    // Ajout du contact dans la liste de la lettre (ordre alphabétique)
    Contact *prec = NULL, *curr = courantLettre->listeContacts;
    Tnatque (curr && comparer_noms_sans_casse(nom, curr->nom) > 0) faire
        prec = curr;
        curr = curr->suivant;
    FTQ
    Si (curr && comparer_noms_sans_casse(nom, curr->nom) == 0)
        Ecrire("Un contact avec ce nom existe deja.\n");
        return;
    Finsi
    Contact* nouveau = creerContact(nom, tel, mail);
    Si(prec == NULL)alors
        nouveau->suivant = courantLettre->listeContacts;
        courantLettre->listeContacts = nouveau;
    Sinon
        prec->suivant = nouveau;
        nouveau->suivant = curr;
    Finsi
    Ecrire("Contact ajoute avec succes.\n");
}
```

2 modifierContact()

fonction de modification d'un contact en utilisant un input de NOM

```
// Fonction pour modifier un contact (modifie tel et/ou mail)
void modifierContact(Lettre* listeLettres, const char* nom, const char* nouveauTel, const char* nouveauMail) {
    Si (nom == NULL || nom[0] == '\0')alors
        printf("Nom invalide.\n");
        return;
    Finsi
    char premiereLettre = toupper(nom[0]);
    Lettre* l = listeLettres;
    // Recherche de la lettre correspondante
    Tantque (l && l->lettre < premiereLettre) Faire
        l = l->lettreSuivante;
    FTQ
    Si (!l || l->lettre != premiereLettre)alors
        printf("Aucun contact trouve avec ce nom.\n");
        return;
    Finsi
    Contact* c = l->listeContacts;
    Tantque (c) Faire
        Si (comparer_noms_sans_casse(nom, c->nom) == 0)alors
            Si (nouveauTel && nouveauTel[0] != '\0')alors
                strncpy(c->tel, nouveauTel, sizeof(c->tel)-1);
                c->tel[sizeof(c->tel)-1] = '\0';
            Finsi
            Si (nouveauMail && nouveauMail[0] != '\0')alors
                strncpy(c->mail, nouveauMail, sizeof(c->mail)-1);
                c->mail[sizeof(c->mail)-1] = '\0';
            Finsi
            Ecrire("Contact modifie avec succes.\n");
            return;
        Finsi
        c = c->suivant;
    FTQ
    Ecrire("Aucun contact trouve avec ce nom.\n");
}
```

3 supprimer contact()

fonction supprimer qui sert a la suppression d'un seul contact

```
// Fonction pour supprimer un contact par nom
void supprimerContact(Lettre** listeLettres, const char* nom) {
    Si (nom == NULL || nom[0] == '\0') alors
        printf("Nom invalide.\n");
        return;
    Finsi
    char premiereLettre = toupper(nom[0]);
    Lettre *precLettre = NULL, *courantLettre = *listeLettres;

    // Recherche de la lettre correspondante
    Tantque (courantLettre && courantLettre->lettre < premiereLettre) Faire
        precLettre = courantLettre;
        courantLettre = courantLettre->lettreSuivante;
    FTQ
    Si(!courantLettre || courantLettre->lettre != premiereLettre)alors
        printf("Aucun contact trouve avec ce nom.\n");
        return;
    Finsi

    // Recherche du contact à supprimer
    Contact *prec = NULL, *curr = courantLettre->listeContacts;
    Tantque (curr && comparer_noms_sans_casse(nom, curr->nom) != 0) faire
        prec = curr;
        curr = curr->suivant;
    FTQ
    Si (!curr) alors
        printf("Aucun contact trouve avec ce nom.\n");
        return;
    Finsi

    // Suppression du contact
    Si (prec == NULL) alors
        courantLettre->listeContacts = curr->suivant;
    Sinon
        prec->suivant = curr->suivant;
    Finsi
    free(curr);
    printf("Contact supprime avec succes.\n");

    // Si la lettre n'a plus de contacts, la supprimer aussi
    Si (courantLettre->listeContacts == NULL) alors
        Si (precLettre == NULL) alors
            *listeLettres = courantLettre->lettreSuivante;
        Sinon
            precLettre->lettreSuivante = courantLettre->lettreSuivante;
        Finsi
        free(courantLettre);
    Finsi
}
```

4 sauvegarderContacts()

```
// Fonction pour sauvegarder tous les contacts dans un fichier texte
void sauvegarderContacts(const Lettre* listeLettres, const char* nomFichier) {
    FILE* f = fopen(nomFichier, "w");
    Si (!f) alors
        Ecrire("Erreur lors de l'ouverture du fichier pour la sauvegarde.\n");
        return;
    }
    const Lettre* l = listeLettres;
    Tantque (l) Faire
        const Contact* c = l->listeContacts;
        Tantque (c) Faire
            fprintf(f, "%s;%s;%s\n", c->nom, c->tel, c->mail);
            c = c->suivant;
        FTQ
        l = l->lettreSuivante;
    FTQ
    fclose(f);
    Ecrire("Contacts sauvegardes dans le fichier '%s'.\n", nomFichier);
}
```

5 rechargerContacts()

fonction qui recharge tous les contacts existant dans un fichier et qui ecrase tous les contacts existant

```
279
280  // Fonction pour recharger les contacts depuis un fichier texte
281 void rechargerContacts(Lettre** listeLettres, const char* nomFichier) {
282      libererContacts(listeLettres);
283      FILE* f = fopen(nomFichier, "r");
284      Si (!f) alors
285          Ecrire("Erreur lors de l'ouverture du fichier pour le rechargement.\n");
286          return;
287      Finsi
288      char ligne[256];
289      Tantque (fgets(ligne, sizeof(ligne), f)) alors
290          char nom[100], tel[20], mail[100];
291          // Suppression du saut de ligne
292          ligne[strcspn(ligne, "\r\n")] = 0;
293          Si (sscanf(ligne, "%99[^;];%19[^;];%99[^\n]", nom, tel, mail) == 3) alors
294              ajouterContact(listeLettres, nom, tel, mail);
295          Finsi
296      FTQ
297      fclose(f);
298      Ecrire("Contacts recharges depuis le fichier '%s'.\n", nomFichier);
299  }
300
```

6 AFFICHAGE DES CONTACTS

fonction d'affichage de tous les contacts deja ajouter

```
111 // Fonction pour afficher tous les contacts par ordre alphabétique
112 void afficherContacts(const Lettre* listeLettres) {
113     int vide = 1;
114     const Lettre* l = listeLettres;
115     Tantque (l) Faire
116         Si (l->listeContacts) alors
117             vide = 0;
118             break;
119         Finsi
120         l = l->lettreSuivante;
121     FTQ
122     Si (vide) alors
123         Ecrire("Aucun contact a afficher.\n");
124         return;
125     Finsi
126     Ecrire("Nom            | Tel           | Mail\n");
127     Ecrire("---------------+---------------+----------------------------\n");
128     l = listeLettres;
129     Tantque (l) Faire
130         const Contact* c = l->listeContacts;
131         Tantque (c) Faire
132             printf("%-13s | %-13s | %-30s\n", c->nom, c->tel, c->mail);
133             c = c->suivant;
134         FTQ
135         l = l->lettreSuivante;
136     FTQ
137 }
138
```

4. JEUX D"ESSAI _

```
--- Gestionnaire de contacts ---
1. Ajouter un contact
2. Supprimer un contact
3. Modifier un contact
4. Afficher tous les contacts
5. Chercher un contact
6. Sauvegarder les contacts dans un fichier
7. Recharger les contacts depuis un fichier
0. Quitter
Votre choix :
```

Maintenant, nous allons simuler les actions de l'utilisateur pour chaque option du menu afin de vérifier que le programme réagit correctement et affiche les résultats attendus.

```
--- Gestionnaire de contacts ---
1. Ajouter un contact
2. Supprimer un contact
3. Modifier un contact
4. Afficher tous les contacts
5. Chercher un contact
6. Sauvegarder les contacts dans un fichier
7. Recharger les contacts depuis un fichier
0. Quitter
Votre choix : 4
Nom             | Tel              | Mail
----------------+------------------+-----------------------------
ahmed           | 0563672632       | ahmed@enim
aymane          | 06327622         | aymane@enim
mohammed        | 056757362        | mohammed@enim
nassr           | 065675372        | nassr@enim
oussama         | 0667382738       | oussama@enim
Voulez-vous revenir au menu ? (o/n) :
```

Afficher tous les contacts

```
Votre choix : 3
Nom du contact a modifier : ahmed
Nouveau telephone (laisser vide pour ne pas changer) :
Nouvel email (laisser vide pour ne pas changer) : Ahmed1@enim
Contact modifie avec succes.
Voulez-vous revenir au menu ? (o/n) :
```

modifier un contact

```
Votre choix : 2
Nom du contact a supprimer : ahmed
Contact supprime avec succes.
Voulez-vous revenir au menu ? (o/n) :
```

supprimer un contact

```
Votre choix : 5
Critere de recherche (nom complet, 1ere lettre ou 1ere+2eme lettre) : a
Nom           Votre choix : 6
              Nom du fichier pour sauvegarder : fichier 1
------------  Contacts sauvegardes dans le fichier 'fichier 1'.
ahmed         Voulez-vous revenir au menu ? (o/n) :
aymane          | 06327622       | aymane@enim
Voulez-vous revenir au menu ? (o/n) :
```

supprimer un contact

```
e choix : 6
du fichier pour sauvegarder : fichier 1
acts sauvegardes dans le fichier 'fichier 1'.
ez-vous revenir au menu ? (o/n) :
```

sauvgarder fichier

```
0. Quitter
Votre choix : 1
Nom : amine
telephone : 07645545467
Email : amine@enim
Contact ajoute avec succes.
Voulez-vous revenir au menu ? (o/n) :
```

ajout d’un contact

```
ayoub;056363222;ayoub@enim
anas;06327622232;anas@enim
mina;0567573012;mina@enim
sara;0656753073;sara@enim
omar;066738405;omar@enim
```

fichier a recharger

```
Votre choix : 7
Nom du fichier a recharger : fichier 2
Contact ajoute avec succes.
Contact ajoute avec succes.
Contact ajoute avec succes.
Contact ajoute avec succes.
Contact ajoute avec succes.
Contacts recharges depuis le fichier 'fichier 2'.
Voulez-vous revenir au menu ? (o/n) : o

--- Gestionnaire de contacts ---
1. Ajouter un contact
2. Supprimer un contact
3. Modifier un contact
4. Afficher tous les contacts
5. Chercher un contact
6. Sauvegarder les contacts dans un fichier
7. Recharger les contacts depuis un fichier
0. Quitter
Votre choix : 4
Nom            | Tel            | Mail
---------------+----------------+-----------------------------
ahmed          | 0563672632     | ahmed@enim
aymane         | 06327622       | aymane@enim
mohammed       | 056757362      | mohammed@enim
nassr          | 065675372      | nassr@enim
oussama        | 0667382738     | oussama@enim
Voulez-vous revenir au menu ? (o/n) :
```

recharge et affichage du fichier

7.LES ATOUTS UNIQUES DE NOTRE CODE

1 Copier quelques option qui ce trouve dans le systeme d'exploitation IOS (Iphone)

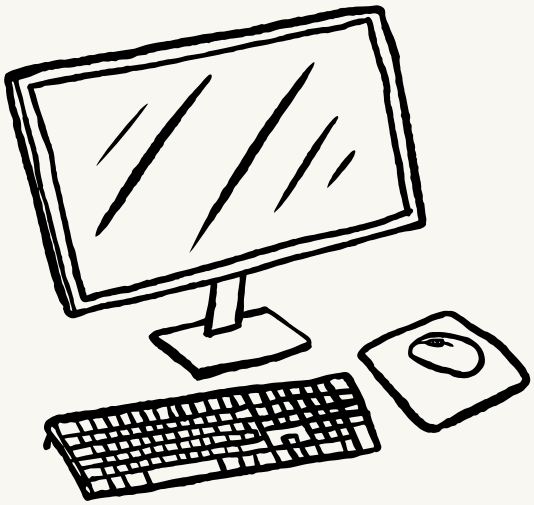
l'ajout des contacts a un critere d'unicite de nom.
Si on ajoute un nom qui existe deja un message s'affiche qui nous informe que un contact precedent existe avec ce nom.

2 La fonction recharge en memoire

notre fonction recharge en memoire ecrase les anciens contacts et ajoute les nouveaux contacts du fichier recharger.

3 L'affichage des contacts

lors de l'affichage d nos contacts on aura tojours la premiere colonne (NOM; | NUM: | EMAIL:) apres vient les contacts un par un



6. CONCLUSION_

Ce projet de gestion de contacts en C a été une expérience extrêmement enrichissante, tant sur le plan technique que personnel. Il nous a offert une première vision concrète du développement logiciel, en nous permettant d'appliquer les concepts théoriques du langage C à un projet réel et fonctionnel.

Grâce à ce travail, nous avons consolidé nos compétences en programmation (gestion de mémoire, structures de données, manipulation de fichiers, etc.), mais aussi développé des soft skills essentiels :

- **Collaboration et travail d'équipe : Répartition des tâches, coordination et résolution collective des problèmes.**
- **Gestion du temps et organisation : Respect des deadlines et adaptation aux imprévus.**
- **Résolution de problèmes : Débogage, optimisation et recherche de solutions techniques.**
- **Communication : Présentation claire du code et des fonctionnalités.**

En somme, ce projet a été une étape clé dans notre apprentissage, renforçant à la fois notre maîtrise du C et notre capacité à travailler efficacement en équipe. Nous sommes fiers du résultat obtenu et motivés à relever de nouveaux défis technologiques !

Un grand merci à notre professeur pour son accompagnement tout au long de ce projet.