Rapport du projet SDD

Gestion des contacts téléphoniques sous forme arborescente

2020/2021

Réalisé par:

FARAH Manal
ATMANI Houda
FARSSI Rahma
HANDI Kaoutar
MAKHCHOUN Khadija
OUZOUGAGH Chaimaa

Encadré par:

Mme.OUAZZANI Khadija

Table des matières

Introduction

Au cours des deux dernières décennies, la prise de contacts téléphoniques était manuelle : rédaction sur papier. Ainsi ceci a engendré pas mal de problèmes de gestion du temps et des contacts en général.

Aujourd'hui avec l'évolution technologique, les répertoires téléphoniques sont devenus purement numériques tout en accordant aux utilisateurs un accès facile et rapide, grâce aux structures de données comme l'arbre qui est une structure dynamique d'éléments appelés aussi parfois « sommet » ou « Nœud ». Ses nœuds sont organisés d'une manière hiérarchique. Il est composé d'un nœud particulier appelé racine ; de plusieurs nœuds intermédiaires possédant chacun un et un seul nœud appelé père, et des nœuds possédant éventuellement un ou plusieurs fils. Les nœuds qui ne possèdent pas de fils sont appelés feuilles.

Un arbre n-aire est utile dans le cas de la représentation d'un dictionnaire et ainsi dans l'accélération de la recherche.

Dans ce qui suit, on présentera tout d'abord le cadre du projet en traitant son cahier des charges fonctionnel, ensuite on va spécifier les algorithmes des fonctions principales du système pour aboutir enfin à la réalisation concrète de ce programme à travers des jeux d'essai.

1. L'objectif du projet

La gestion du contact téléphonique désigne l'organisation de l'ensemble des contacts à savoir : l'ajout, la suppression, la sauvegarde, la recherche d'un contact.

Un tel programme s'avère très utile de nos jours, notamment si nous devons gérer des contacts multiples, pour sauvegarder les coordonnées de nos contacts et y avoir recours en au moment voulu.

Dans ce contexte s'inscrit le but de notre projet à savoir la création d'une application de gestion des contacts téléphonique en se basant sur une structure arborescente.

2. Description des fonctionnalités du programme

Le système propose un menu dont les fonctionnalités attendues sont :

- Ajout d'un nouveau contact.
- Suppression d'un contact existant.
- Recherche de contacts par leurs noms : L'utilisateur pourra saisir la première, ou les deux premières lettres ou le nom complet du contact souhaité. La recherche fournira la liste des contacts qui vérifient ces critères.
- Affichage de la liste des contacts.
- Quitter : Arrêt de l'exécution du programme

3. Les structures de données utilisées

L'implémentation est basée sur les **arbres n-aires**.

On rappelle que l'arbre n-aire est une structure de données dynamique d'éléments organisés d'une manière hiérarchique appelés « nœuds ». Il est composé d'un nœud particulier appelé « racine », de plusieurs nœuds intermédiaires possédant chacun un et un seul nœud appelé père et un ou plusieurs nœud appelés fils. Les nœuds qui ne possèdent pas de fils sont appelés feuilles.

Dans notre application la racine est une structure de donnée appelée contact qui se compose de 3 champs :

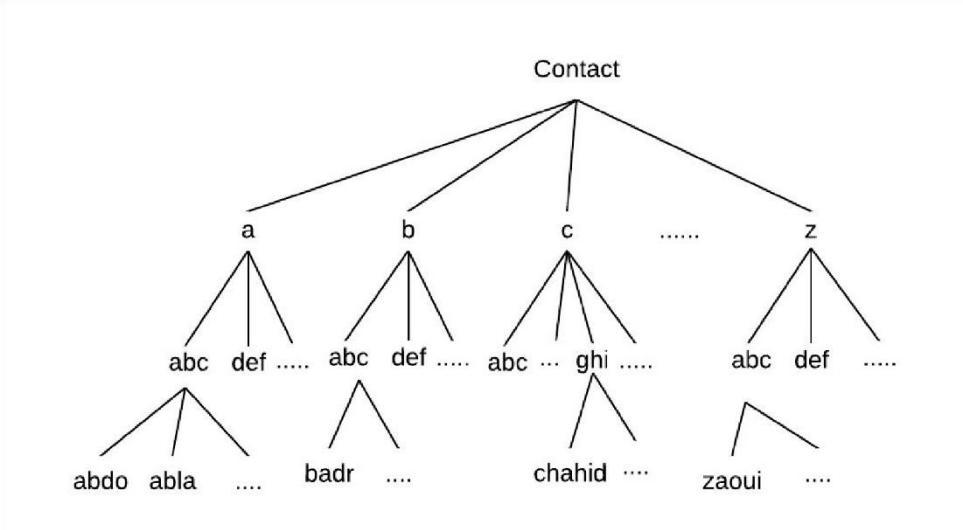
- Info qui est à son tour une structure de donnée de type contact comportant trois champs (le nom, le téléphone et le mail)
- Fils qui est un pointeur vers le premier fils de la racine contact
- Frère qui est un pointeur nul dans ce cas vu que la racine n'a pas de frère

Les nœuds de 2eme niveau à savoir les fils de la racine sont les alphabets. Chacun a comme fils un trio et comme frère un alphabet

Les nœuds de 3eme niveau à savoir les fils de la racine sont des trios des alphabets. Chacun a comme fils un nom de contact et comme frère un autre trio

Les feuilles de cet arbre sont les noms de contact, chacune pointe vers un frère qui est un nom de contact et ils ont ne possèdent pas de fils.

FIGURE 1 *Modèle illustrant l'arbre n-aire du Contact*



4. Le langage utilisé

L'implémentation du programme s'est appuyé sur le **langage C** qui est un langage de programmation impératif conçu pour la programmation système, tout en le compilant à l'aide de l'**IDE** (Environnement de développement intégré) **Code::Blocks**.

5. Les algorithmes principaux

☞ **Notations utilisées :**

trio: chaîne de caractères qui représente l'ensemble ordonné des 3 à 4 alphabets "abc","def","ghi,"jkl","mno",pqr","stuv","wxyz"

a. Déclaration des structures

main.c X

```c
/*********************************************************************
  Programme de gestion d'un repertoire telephonique
*********************************************************************/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef struct mon_contact{
    char nom[16];
    char tel[11];
    char mail[26];
} Contact;

struct elem_arbre {
    Contact info;
    struct elem_arbre *fils;
    struct elem_arbre *frere;
};
```

b. Fonction d'ajout

C'est la fonction qui sert à ajouter un contact dans notre arbre par ordre (la première lettre et le trio du contact sera ajouter automatiquement)

main.c

```
138 struct elem_arbre * ajout(struct elem_arbre *p, Contact e) {
139     char lettre0_nom[2];
140     lettre0_nom[0] = e.nom[0];
141     lettre0_nom[1] = '\0';
142     char *tr=find_trio(e);
143     if(p->fils==NULL){
144
145     //initialisation des lettres et trios et contacts
146         struct elem_arbre *element = NULL;
147         element=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
148         element->info=e;
149         element->fils=NULL;
150         element->frere=NULL;
151
152
153         struct elem_arbre *trio = NULL;
154         trio=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
155             Contact rt2={NULL,NULL,NULL};
156         strcpy(rt2.nom,tr);
157         trio->info=rt2;
158         trio->fils=element;
159         trio->frere=NULL;
160
161         struct elem_arbre *lettre_1 = NULL;
162         lettre_1=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
163             Contact rt={NULL,NULL,NULL};
164         strcpy(rt.nom,lettre0_nom);
165         lettre_1->info=rt;
166         lettre_1->fils=trio;
167          lettre_1->frere=NULL;
168
169          p->fils=lettre_1;
170
171     }
172     else{
173
174
175             if (strcmp(lettre0_nom, p->fils->info.nom)< 0){
176
177             struct elem_arbre *element = NULL;
178             element=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
179             element->info=e;
180             element->fils=NULL;
181             element->frere=NULL;
182
183             struct elem_arbre *trio = NULL;
184             trio=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
185             Contact rt2={NULL,NULL,NULL};
186             strcpy(rt2.nom,tr);
187             trio->info=rt2;
188             trio->fils=element;
189             trio->frere=NULL;
190
191             struct elem_arbre *lettre_1 = NULL;
192             lettre_1=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
193             Contact rt={NULL,NULL,NULL};
194             strcpy(rt.nom,lettre0_nom);
195             lettre_1->info=rt;
```

main.c

```
196             lettre_1->fils=trio;
197             lettre_1->frere=p->fils;
198
199             p->fils=lettre_1;
200         }
201
202         else if (strcmp(lettre0_nom, p->fils->info.nom)== 0){
203             ajout_trio (p->fils, e,tr) ;
204         }
205         else if (p->fils->frere== NULL){
206
207             struct elem_arbre *element = NULL;
208             element=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
209             element->info=e;
210             element->fils=NULL;
211             element->frere=NULL;
212
213             struct elem_arbre *trio = NULL;
214             trio=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
215             Contact rt2={NULL,NULL,NULL};
216             strcpy(rt2.nom,tr);
217             trio->info=rt2;
218             trio->fils=element;
219             trio->frere=NULL;
220
221             struct elem_arbre *lettre_1 = NULL;
222             lettre_1=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
223             Contact rt={NULL,NULL,NULL};
224             strcpy(rt.nom,lettre0_nom);
225             lettre_1->info=rt;
226             lettre_1->fils=trio;
227             lettre_1->frere=NULL;
228
229             p->fils->frere=lettre_1;
230         }
231         else {
232             struct elem_arbre *element_0 = p->fils;
233             while (element_0->frere!=NULL){
234                 if (strcmp(lettre0_nom, element_0->frere->info.nom)< 0){
235
236                     struct elem_arbre *element = NULL;
237                     element=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
238                     element->info=e;
239                     element->fils=NULL;
240                     element->frere=NULL;
241
242                     struct elem_arbre *trio = NULL;
243                     trio=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
244                     Contact rt2={NULL,NULL,NULL};
245                     strcpy(rt2.nom,tr);
246                     trio->info=rt2;
247                     trio->fils=element;
248                     trio->frere=NULL;
249
```

main.c

```
250                     struct elem_arbre *lettre_1 = NULL;
251                     lettre_1=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
252                     Contact rt={NULL,NULL,NULL};
253                     strcpy(rt.nom,lettre0_nom);
254                     lettre_1->info=rt;
255                     lettre_1->fils=trio;
256                     lettre_1->frere=element_0->frere;
257
258                     element_0->frere=lettre_1;
259
260                     break;
261                 }
262                 else if (strcmp(lettre0_nom, element_0->frere->info.nom)== 0){
263                     ajout_trio (element_0->frere, e,tr) ;
264                     break;
265                 }
266                 element_0=element_0->frere;
267
268                 if(element_0->frere==NULL){
269
270                     struct elem_arbre *element = NULL;
271                     element=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
272                     element->info=e;
273                     element->fils=NULL;
274                     element->frere=NULL;
275
276                         struct elem_arbre *trio = NULL;
277                         trio=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
278                         Contact rt2={NULL,NULL,NULL};
279                         strcpy(rt2.nom,tr);
280                         trio->info=rt2;
281                         trio->fils=element;
282                         trio->frere=NULL;
283
284                         struct elem_arbre *lettre_1 = NULL;
285                         lettre_1=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
286                         Contact rt={NULL,NULL,NULL};
287                         strcpy(rt.nom,lettre0_nom);
288                         lettre_1->info=rt;
289                         lettre_1->fils=trio;
290                         lettre_1->frere=element_0->frere;
291
292                         element_0->frere=lettre_1;
293                         break;
294                     }
295                 }
296             }
297         }
298     return p;
299
300 }
301
```

ajout trio : se charger d'ajouter le trio adapté au contact

*main.c

```
305 void ajout_trio (struct elem_arbre *r, Contact e,char * trio_lettre) {
306     struct elem_arbre *trio = r->fils;
307     if (strcmp(trio_lettre, trio->info.nom)< 0){
308         struct elem_arbre *element = NULL;
309         element=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
310         element->info=e;
311         element->fils=NULL;
312         element->frere=NULL;
313 
314         struct elem_arbre *new_trio = NULL;
315         new_trio=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
316             Contact rt={NULL,NULL,NULL};
317         strcpy(rt.nom,trio_lettre);
318         new_trio->info=rt;
319         new_trio->fils=element;
320         new_trio->frere=r->fils;
321         r->fils=new_trio;
322     }
323     else if (strcmp(trio_lettre, trio->info.nom)== 0){
324         ajout_contact(trio, e);
325     }
326     else if(trio->frere==NULL){
327         struct elem_arbre *element = NULL;
328         element=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
329         element->info=e;
330         element->fils=NULL;
331         element->frere=NULL;
```

```
333         struct elem_arbre *new_trio = NULL;
334         new_trio=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
335             Contact rt={NULL,NULL,NULL};
336         strcpy(rt.nom,trio_lettre);
337         new_trio->info=rt;
338         new_trio->fils=element;
339         new_trio->frere=NULL;
340         trio->frere=new_trio;
341     }
342     else{
343         while(trio->frere!=NULL){
344             if (strcmp(trio_lettre, trio->frere->info.nom)< 0){
345                 struct elem_arbre *element = NULL;
346                 element=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
347                 element->info=e;
348                 element->fils=NULL;
349                 element->frere=NULL;
350 
351                 struct elem_arbre *new_trio = NULL;
352                 new_trio=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
353                     Contact rt={NULL,NULL,NULL};
354                 strcpy(rt.nom,trio_lettre);
355                 new_trio->info=rt;
356                 new_trio->fils=element;
357                 new_trio->frere=trio->frere;
358                 trio->frere=new_trio;
359                 break;
360             }
```

*main.c

```
361             else if (strcmp(trio_lettre, trio->frere->info.nom)== 0){
362                 ajout_contact(trio, e);
363                 break;
364             }
365             trio=trio->frere;
366             if(trio->frere==NULL){
367 
368                 struct elem_arbre *element = NULL;
369                 element=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
370                 element->info=e;
371                 element->fils=NULL;
372                 element->frere=NULL;
373 
374                 struct elem_arbre *new_trio = NULL;
375                 new_trio=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
376                     Contact rt2={NULL,NULL,NULL};
377                 strcpy(rt2.nom,trio_lettre);
378                 new_trio->info=rt2;
379                 new_trio->fils=element;
380                 new_trio->frere=NULL;
381                 trio->frere=new_trio;
382                 break;
383             }
384         }
385     }
386 }
```

- ✓ **Création du contact qui sera ajouter à l'arbre**

*main.c

```
389 void ajout_contact (struct elem_arbre *r, Contact e) {
390 
391     struct elem_arbre *contact = r->fils;
392     if (strcmp(e.nom, contact->info.nom)< 0){
393         struct elem_arbre *element = NULL;
394         element=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
395         element->info=e;
396         element->fils=NULL;
397         element->frere=contact;
398 
399 
400         r->fils=element;
401     }
402     else if(contact->frere==NULL){
403 
404         struct elem_arbre *element = NULL;
405         element=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
406         element->info=e;
407         element->fils=NULL;
408         element->frere=NULL;
409 
410 
411         contact->frere=element;
412     }
413     else{
414         while(contact->frere!=NULL){
415             if (strcmp(e.nom, contact->frere->info.nom)< 0){
416                 struct elem_arbre *element = NULL;
417                 element=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
```

```
417                 element=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
418                 element->info=e;
419                 element->fils=NULL;
420                 element->frere=contact->frere;
421
422                 contact->frere=element;
423
424                 break;
425             }
426             contact=contact->frere;
427
428             if(contact->frere==NULL){
429
430                 struct elem_arbre *element = NULL;
431                 element=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
432                 element->info=e;
433                 element->fils=NULL;
434                 element->frere=NULL;
435
436                 contact->frere=element;
437                 break;
438
439             }
440         }
441     }
442 }
```

- ✓ **Cette fonction nous permet de déterminer le trio auquel appartient la deuxième lettre du nom du contact**

*main.c

```
444
445 char * find_trio(Contact e){
446     char T[8][5]={"abc\0","def\0","ghi\0","jkl\0","mno\0","pqr\0","stuv\0","wxyz\0"};
447     char l=e.nom[1];
448     char *tr=(char*)malloc(4*sizeof(char));
449     int i;
450     for(i=1;i<8;i++){
451     if(T[i][0]>l){
452         strcpy(tr,T[i-1]);
453         return tr;
454     }
455     }
456     strcpy(tr,T[i-1]);
457     return tr;
458 }
```

c. Afficher contact

*main.c

```
459
460 void afficher(struct elem_arbre *r){
461     struct elem_arbre *letters = r->fils;
462     while (letters!=NULL){
463         printf("Lettre : %s\n",letters->info.nom);
464         struct elem_arbre *trios = letters->fils;
465         while (trios!=NULL){
466             printf("\tTrio : %s\n",trios->info.nom);
467             struct elem_arbre *contacts = trios->fils;
468             while (contacts!=NULL){
469                 printf("\t\t %s %s %s\n",contacts->info.nom,contacts->info.tel,contacts->info.mail);
470                 contacts=contacts->frere;
471             }
472             trios=trios->frere;
473         }
474         letters=letters->frere;
475     }
476
477 }
```

d. Supprimer contact

*main.c

```
478
479 void supprimer_contact (struct elem_arbre *r, char l[]){
480     struct elem_arbre *lettres = r->fils;
481     struct elem_arbre *lettres_2 = lettres;
482     while(l[0]>lettres->info.nom[0] && lettres->frere!=NULL){
483
484         lettres_2=lettres;
485         lettres=lettres->frere;
486     }
487
488     if (l[0]!=lettres->info.nom[0]){
489         printf("le nom saisi n'existe pas !!! \n");
490     }
491     else{
492         struct elem_arbre *trios = lettres->fils;
493         struct elem_arbre *trios_2 = trios;
494         do{
495             int taille=strlen(trios->info.nom);
496
497             if(trios->info.nom[0]<=l[1] && trios->info.nom[taille-1]>=l[1]){
498                 if (strcmp(l, trios->fils->info.nom)== 0){
499                     if(trios->fils->frere!=NULL){
500                         struct elem_arbre *temp=trios->fils;
501                         trios->fils=trios->fils->frere;
502                         free(temp);
503                     }
504                     else{
505                         if(trios_2 == trios && trios->frere==NULL){
506                             if(lettres_2==lettres && lettres->frere==NULL){
```

*main.c

```
507                                 r->fils=NULL;
508                                 free(lettres);
509                             }
510                             else if(lettres_2==lettres && lettres->frere!=NULL){
511                                 r->fils=lettres->frere;
512                                 free(lettres);
513                             }
514                             else{
515                                 lettres_2->frere=lettres->frere;
516                                 free(lettres);
517                             }
518                         }
519                         else if(trios_2 == trios && trios->frere!=NULL){
520                             lettres->fils=trios->frere;
521                             free(trios);
522                         }
523                         else{
524                             trios_2->frere=trios->frere;
525                             free(trios);
526                         }
527                     }
528                     break;
529                 }
530                 else{
531                         struct elem_arbre *contacts=trios->fils;
532 
533                         while(contacts->frere!=NULL && strcmp(l, contacts->frere->info.nom)!= 0){
534                             contacts=contacts->frere;
535                         }
536                             if(strcmp(l, contacts->frere->info.nom)==0){
537                                 struct elem_arbre *temp=contacts->frere;
538                                 contacts->frere=contacts->frere->frere;
539                                 free(temp);
540                             }
541                             else{
542                                 printf("le nom saisi n'existe pas!! \n");
543                             }
544                             break;
545                     }
546 
547                 }
548                 trios_2 = trios;
549                 trios=trios->frere;
550             }while(trios!=NULL);
551             if(trios==NULL)     printf("le nom saisi n'existe pas 2 !! \n");
552     }
553 }
```

e. Chercher contact

✓ Par première lettre :

*main.c

```
554
555 void chercher_contact_par_1_lettre(struct elem_arbre *r, char l[]){
556     struct elem_arbre *lettres = r->fils;
557     while(l[0]>lettres->info.nom[0] && lettres->frere!=NULL){
558         lettres=lettres->frere;
559     }
560
561     if (l[0]!=lettres->info.nom[0]){
562         printf("Aucun nom ne commance par %s!! \n",l);
563     }
564     else{
565         printf("Les noms commancant pas %s sont: \n",l);
566
567         struct elem_arbre *trios = lettres->fils;
568         while (trios!=NULL){
569
570             struct elem_arbre *contacts = trios->fils;
571             while (contacts!=NULL){
572                 printf("\t- Contact : %s %s %s\n",contacts->info.nom,contacts->info.tel,contacts->info.mail);
573                 contacts=contacts->frere;
574             }
575             trios=trios->frere;
576         }
577     }
578
579 }
```

✓ Par les deux premières lettres :

*main.c

```
581 void chercher_contact_par_2_lettre(struct elem_arbre *r, char l[]){
582     struct elem_arbre *lettres = r->fils;
583     while(l[0]>lettres->info.nom[0] && lettres->frere!=NULL){
584         lettres=lettres->frere;
585     }
586     if (l[0]!=lettres->info.nom[0]){
587         printf("Aucun nom ne commance par %s!! \n",l);
588     }
589     else{
590         struct elem_arbre *trios = lettres->fils;
591         do{
592             int taille=strlen(trios->info.nom);
593
594             if(trios->info.nom[0]<=l[1] && trios->info.nom[taille-1]>=l[1]){
595                 printf("Les noms commancant pas %s sont: \n",l);
596                 struct elem_arbre *contacts = trios->fils;
597                 while(contacts!=NULL && contacts->info.nom[1]<=l[1]){
598                     if(contacts->info.nom[1]==l[1]){
599                         printf("\t- Contact : %s %s %s\n",contacts->info.nom,contacts->info.tel,contacts->info.mail);
600                     }
601                     contacts=contacts->frere;
602                 }
603                 break;
604             }
605             trios=trios->frere;
606         }while(trios!=NULL);
607         if(trios==NULL){        printf("Aucun nom ne commance par %s!! \n",l);  }
608     }
609 }
```

✓ Par nom :

*main.c

```
611 struct elem_arbre * chercher_contact_par_nom(struct elem_arbre *r, char l[]){
612     struct elem_arbre *lettres = r->fils;
613     while(l[0]>lettres->info.nom[0] && lettres->frere!=NULL){
614         lettres=lettres->frere;
615     }
616     if (l[0]!=lettres->info.nom[0]){
617         printf("Aucun contact avec le nom %s!! \n",l);
618     }
619     else{
620         struct elem_arbre *trios = lettres->fils;
621         do{
622             int taille=strlen(trios->info.nom);
623             if(trios->info.nom[0]<=l[1] && trios->info.nom[taille-1]>=l[1]){
624                 struct elem_arbre *contacts = trios->fils;
625                 while(contacts!=NULL && strcmp(contacts->info.nom, l)< 0){
626                     contacts=contacts->frere;
627                 }
628                 if(contacts!=NULL && strcmp(contacts->info.nom, l)== 0){
629                     printf("\tContact existant : %s %s %s\n",contacts->info.nom,contacts->info.tel,contacts->info.mail);
630                     return contacts;
631                 }
632                 else        printf("Aucun contact avec le nom %s!! \n",l);
633                 break;
634             }
635             trios=trios->frere;
636         }while(trios!=NULL);
637         if(trios==NULL){        printf("Aucun contact avec le nom %s!! \n",l);  }
638     }
639     return NULL;
640 }
```

6. Jeu d’essai

a. Code main

```c
int main(void)
{
    int choix = 0;
    Contact e;
    struct elem_arbre * p = NULL;
        Contact rt={"contact",NULL,NULL};
    p=(struct elem_arbre*)(malloc(sizeof(struct elem_arbre)));
        p->info=rt;
        p->fils=NULL;
        p->frere=NULL;
    char ln[16];
/*Afficher le menu à l'utilisateur*/
    system("cls");
    system("COLOR 3F");
    system("COLOR 3f");
    system("cls");
    int choix1;
    printf("\n\n");
    printf("\n\t\t\t\t\t\t      Bonjour (^_^) !!\n\t\t\t Voici un programme qui permet de gerer votre repertoire telephonique\n ");
    printf("\n                          ------------------Ce programme a ete realise par:----------------\n");
    printf("                                |                      * HANDI Kaoutar                          |\n");
    printf("                                |                      * MAKHCHOUN Khadija                      |\n");
    printf("                                |                      * ATMANI Houda                           |\n");
    printf("                                |                      * FARAH Manal                            |\n");
    printf("                                |                      * OUZOUGAGH Chaimae                      |\n");
    printf("                                |                      * FARSSI Rahma                           |\n");
    printf("                               ------------------------------Encadre par :-------------------------\n");
    printf("                                |                  Mme.TOUHAMI OUAZZANI Khadija                 |\n");
    printf("                                 ------------------------------------------------------------\n\n");

    printf("                                    ********************************************************\n");
    printf("                                    |                REPERTOIRE TELEPHONIQUE               |\n");
    printf("                                    |                     ~~ MENU ~~                       |\n");
```

```c
    printf("                                    |                REPERTOIRE TELEPHONIQUE               |\n");
    printf("                                    |                     ~~ MENU ~~                       |\n");
    printf("                                    ********************************************************\n");
    printf("                                    |             1 . Ajouter un contact                   |\n");
    printf("                                    |             2 . Supprimer un contact                 |\n");
    printf("                                    |             3 . Chercher un contact                   |\n");
    printf("                                    |             4 . Afficher le repertoire courant       |\n");
    printf("                                    |             5 . Quitter                              |\n");
    printf("                                    ********************************************************\n");
/*Revenir au menu après chaque instruction jusqu'à ce que l'utilisateur quitte*/
    while (choix != 6) {
        /*Obtenir l'option de l'utilisateur*/
        printf("\nVeuillez selectionner une option: ");
        scanf("%d", &choix);
        /*Si l'option est 1 (Ajouter):*/
        if (choix == 1) {
                /*Prendre les données de l'utilisateur.*/
            printf("Veuillez saisir votre nom: ");
            scanf("%s", &e.nom);
            printf("Veuillez saisir votre numero de telephone: ");
            scanf("%s", &e.tel);
            printf("Veuillez saisir votre adresse mail: ");
            scanf("%s", &e.mail);
            /*Créez un nouveau noeud*/
            p = ajout(p, e);
            /*Confirmez la création du noeud*/
            printf("------Contact enregistre-----\n\n");

        }
        else if (choix == 2) {

            printf("Veuillez saisir le nom du contact: ");
            scanf("%s", &ln);
```

```c
        else if (choix == 2) {

            printf("Veuillez saisir le nom du contact: ");
            scanf("%s", &ln);
            supprimer_contact(p, ln);
        }
        else if (choix == 3) {

            printf("Veuillez saisir le nom du contact: ");
            scanf("%s", &ln);
            if(strlen(ln)==1)
                chercher_contact_par_1_lettre(p, ln);
            else if(strlen(ln)==2)
                chercher_contact_par_2_lettre(p, ln);
            else{   chercher_contact_par_nom(p, ln);}
        }
        else if (choix == 4) {
            afficher(p);
        }
        else if (choix == 5) {
            printf("\n\nMerci pour avoir utilise ce programme (^_^) a tres bientot !!\n");
            break;
        }
    /*Si l'utilisateur ne choisie pas une option existante*/
        else {
            /*afficher un message d'erreur.*/
            printf("Ce choix n'existe pas, choisir un numero de 1 a 6\n\n");
            break;
        }
    }
    return 0;
}
```

b. Introduction et menu principal

Cette interface affiche à l'utilisateur un message indiquant la mission du programme et ses créateurs. Ainsi Le menu est affiché comme suit :

```
                              Bonjour (^_^) !!
                  Voici un programme qui permet de gerer votre repertoire telephonique

                     ------------------Ce programme a ete realise par:----------------
                     |                      * HANDI Kaoutar                          |
                     |                      * MAKHCHOUN Khadija                      |
                     |                      * ATMANI Houda                           |
                     |                      * FARAH Manal                            |
                     |                      * OUZOUGAGH Chaimae                      |
                     |                      * FARSSI Rahma                           |
                     ----------------------------Encadre par :-------------------------
                     |                   Mme.TOUHAMI OUAZZANI Khadija                |
                      ------------------------------------------------------------

                         ******************************************************
                         |               REPERTOIRE TELEPHONIQUE              |
                         |                     ~~ MENU ~~                     |
                         ******************************************************
                         |             1 . Ajouter un contact                 |
                         |             2 . Supprimer un contact               |
                         |             3 . Chercher un contact                |
                         |             4 . Afficher le repertoire courant     |
                         |             5 . Quitter                            |
                         ******************************************************

Veuillez selectionner une option:
```

Pour accéder aux fonctionnalités proposées dans le menu, l'utilisateur doit taper un nombre compris entre 1 et 5 sinon un message d'erreur s'affiche.

c. Ajout de contact

Le choix de l'option **« 1 »** permet à l'utilisateur d'ajouter un contact en saisissant son nom, son numéro de téléphone et son adresse mail. Ainsi un message s'affiche pour confirmer que l'ajout du contact est effectué avec succès.

```
Veuillez saisir votre nom: amin
Veuillez saisir votre numero de telephone: 02314568
Veuillez saisir votre adresse mail: amin@gmail.com
------Contact enregistre-----


Veuillez selectionner une option: 1
Veuillez saisir votre nom: chakib
Veuillez saisir votre numero de telephone: 032569874
Veuillez saisir votre adresse mail: chakib@gmail.com
------Contact enregistre-----


Veuillez selectionner une option: 1
Veuillez saisir votre nom: zina
Veuillez saisir votre numero de telephone: 086723457
Veuillez saisir votre adresse mail: zina.lehbila@gmail.com
------Contact enregistre-----


Veuillez selectionner une option: 1
Veuillez saisir votre nom: zaki
Veuillez saisir votre numero de telephone: 094567231
Veuillez saisir votre adresse mail: zaki.hacker@gmail.com
------Contact enregistre-----


Veuillez selectionner une option: 1
Veuillez saisir votre nom: chakira
Veuillez saisir votre numero de telephone: 098111446
Veuillez saisir votre adresse mail: waka.waka@gmail.com
------Contact enregistre-----


Veuillez selectionner une option:
```

d. Affichage de la liste des contacts

Le choix de l'option **« 4 »** effectue l'affichage de l'intégralité du répertoire téléphonique trié alphabétiquement par le nom du contact.

```
Veuillez selectionner une option: 4
Lettre : a
        Trio : mno
                Contact : amin 02314568 amin@gmail.com
Lettre : c
        Trio : ghi
                Contact : chakib 032569874 chakib@gmail.com
                Contact : chakira 098111446 waka.waka@gmail.com
Lettre : z
        Trio : abc
                Contact : zaki 094567231 zaki.hacker@gmail.com
        Trio : ghi
                Contact : zina 086723457 zina.lehbila@gmail.com

Veuillez selectionner une option:
```

e. Recherche de contacts

Le choix de l'option « 3 » permet à l'utilisateur de chercher un contact souhaité en saisissant le nom du contact ou bien les lettres premières constituant le nom de ce contact. Ainsi la recherche s'effectue comme suit:

f. Supprimer un contact

Le choix de l'option « 2 » permet à l'utilisateur de supprimer un contact choisi préalablement. Une fois le contact souhaité est supprimé on peut constater par l'affichage que ce dernier est bel bien supprimé

```
Veuillez selectionner une option: 4
Lettre : c
        Trio : ghi
                 : chakib 032569874 chakibhjdgjt@mail.com
                 : chakira 02365874 defgthyjupoju@mail.com
Lettre : z
        Trio : abc
                 : zaki 0231456987 llfgshktpr@mail.coom
        Trio : ghi
                 : zina 012365897 nhkmfhzima@mail.com

Veuillez selectionner une option: 2
Veuillez saisir le nom du contact: zaki

Veuillez selectionner une option: 2
Veuillez saisir le nom du contact: chakib

Veuillez selectionner une option: 4
Lettre : c
        Trio : ghi
                 : chakira 02365874 defgthyjupoju@mail.com
Lettre : z
        Trio : ghi
                 : zina 012365897 nhkmfhzima@mail.com

Veuillez selectionner une option:
```

g. Quitter

Pour quitter le programme, l'utilisateur doit choisir l'option **« 5 »** puis un message s'affiche pour confirmer l'action.

```
Veuillez selectionner une option: 5


Merci pour avoir utilise ce programme (^_^) a tres bientot !!

Process returned 0 (0x0)   execution time : 13.825 s
Press any key to continue.
```

conclusion

Pour mener à bien ce projet, nous avons dû enrichir et approfondir nos connaissances autant du point de vue des structures des données notamment les arbres comme une structure dynamique de donnée qui est très favorable dans la gestion des contacts téléphoniques .Ainsi ce projet nous a permet de bien consolider nos accointances dans le langage c.