

October 27, 2024

Sommaire

Introduction	3
1 Principe	4
2 Conception	6
2.1 Génèse du projet	6
2.2 Fonctionnalités à implémenter	7
2.3 Structure de données	7
2.4 Programme Annexe	8
3 Implémentation en C du projet principal:	9
3.1 Getters et Setters	9
3.2 Trouver une Personne	10
3.3 Lien entre personnes	10
3.4 Manipulation fichiers	10
3.5 Modification d'une personne	10
3.6 Gestion d'un arbre	11
3.7 Consulter les éléments d'un arbre	11
3.8 Menu affiché à l'utilisateur	12
4 Implémentation en C du projet annexe:	13
4.1 Projet annexe	13
4.2 Description des fonctions	13
5 Résultats projet annexe	14
5.1 Affichage du Menu:	14
5.2 Choix 1: Ajouter une personne	14
5.3 Choix 2: Marier deux personne	15
5.4 Choix 3: Ajouter un enfant	16
5.5 Choix 4: Afficher la famille	17
5.6 Sauvegarde	17
6 Conclusion	18

Introduction

L'objectif de ce projet est de créer et gérer un ou plusieurs arbres généalogiques. Un arbre généalogique est une représentation graphique de la généalogie ascendante ou descendante d'un individu. Dans notre projet, nous allons chercher à représenter les différents éléments de l'arbre à l'aide d'un programme en C, qui permettra à l'utilisateur d'effectuer plusieurs actions sur les arbres.

Il s'agit ici de développer nos compétences en C, plus précisément en ce qui concerne les pointeurs, structures et listes chaînées, vues d'un point de vue théorique en cours et TD.

1 Principe

Les premières questions que auxquelles nous pouvons nous intéresser sont celles concernant le passage d'arbre aux données sur les membres de la famille à travers notre programme.

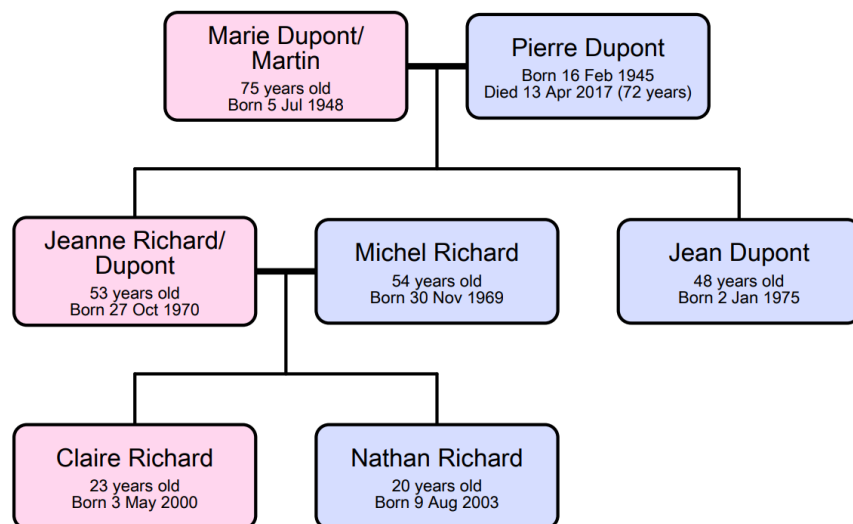


Figure 2: Arbre généalogique de la famille Dupont

Visualisons les différents cas possibles avec cet arbre, selon les membres et leurs liens tout en affichant leur caractéristiques:

Les enfants de Pierre Dupont:

Lien	Informations
Père	Nom: Dupont Prénom : Pierre Date de naissance: 16/02/1945 Vivant: non Date de décès: 13/04/2017 Enfant: Jean Dupont Jeanne Dupont
Fils	Nom: Dupont Prénom : Jean Date de naissance: 02/01/1975 Vivant: oui Père: Pierre Dupont
Fille	Nom: Dupont/Richard Prénom : Jeanne Date de naissance: 27/10/1970 Vivant: oui Père: Pierre Dupont

Lignée maternelle de Claire Richard:

Lien	Informations
Grand-mère	Nom: Dupont/Martin Prénom : Marie Date de naissance: 05/07/1948 Vivant: oui Enfant: Jeanne Dupont
Mère	Nom: Dupont/Richard Prénom : Jeanne Date de naissance: 27/10/1970 Vivant: oui Mère : Marie Dupont/Martin Enfant: Claire Richard
Fille	Nom: Richard Prénom : Claire Date de naissance: 03/05/2000 Vivant: oui Mère: Jeanne Richard/Dupont

Les enfants de Jeanne Dupont:

Lien	Informations
Mère	Nom: Dupont/Richard Prénom : Jeanne Date de naissance: 27/10/1970 Vivant: oui Enfant: Claire Richard Nathan Richard
Fils	Nom: Nathan Prénom : Richard Date de naissance: 09/08/2003 Vivant: oui Mère: Jeanne Richard/Dupont
Fille	Nom: Richard Prénom : Claire Date de naissance: 03/05/2000 Vivant: oui Mère: Jeanne Richard/Dupont

Le couple Richard:

Lien	Informations
Mari	Nom: Richard Prénom : Michel Date de naissance: 30/11/1969 Vivant: oui Conjoint: Jeanne Richard/Dupont
Épouse	Nom: Dupont/Richard Prénom : Jeanne Date de naissance: 27/10/1970 Vivant: oui Coinjoint: Michel Richard

Les neveux de Jean Dupont:

Lien	Informations
Oncle	Nom: Dupont Prénom : Jean Date de naissance: 02/01/1975 Vivant: oui Neveux: Claire Richard Nathan Richard
Neveu	Nom: Nathan Prénom : Richard Date de naissance: 09/08/2003 Vivant: oui Oncle: Jean Dupont
Nièce	Nom: Richard Prénom : Claire Date de naissance: 03/05/2000 Vivant: oui Oncle: Jean Dupont

Ces différents exemples nous permettent de mieux visualiser les résultats qui devront être restitué par le gestionnaire, et l'organisation à suivre.

2 Conception

2.1 Génèse du projet

Ce projet consiste en la création d'un gestionnaire d'arbre généalogique. Il doit permettre avant tout à l'utilisateur de consulter l'arbre généalogique mais aussi de le modifier en cas d'erreur ou de nouveau membre.

Le gestionnaire d'arbre généalogique doit être capable de fournir à l'utilisateur un ensemble de fonctionnalités lui permettant de gérer l'arbre généalogique et/ou de consulter son contenu. Des options d'enregistrement et de chargement à partir d'un fichier ont été ajoutées et seront également disponibles pour l'utilisateur.

2.2 Fonctionnalités à implémenter

Le gestionnaire d'arbre généalogique doit offrir à l'utilisateur un ensemble de fonctionnalités permettant de gérer l'arbre et/ou de consulter son contenu. Les fonctionnalités suivantes sont prévues, avec l'ajout d'options d'enregistrement et de chargement à partir d'un fichier :

- Charger un arbre existant à partir d'un fichier (demander le nom du fichier à l'utilisateur) ;
- Créer un arbre vide ;
- Afficher toutes les personnes de l'arbre (dans un ordre quelconque) ;
- Afficher la descendance d'une personne choisie (demander l'identité de la personne à l'utilisateur) ;
- Afficher l'ascendance d'une personne choisie (demander l'identité de la personne à l'utilisateur) ;
- Afficher le lien entre deux personnes choisies (demander l'identité des personnes à l'utilisateur, répondre par le nom du lien) ;
- Afficher la ou les personne(s) ayant un lien précis avec une autre choisie, par exemple les frères et sœurs, les cousins, les grands-parents... (demander l'identité de la personne à l'utilisateur puis la nature du lien cherché, répondre en affichant les personnes correspondantes) ;
- Gérer l'arbre en ajoutant, modifiant (attributs et liens parentaux) et supprimant des personnes ;
- Supprimer tout le contenu de l'arbre (vider l'arbre).

Il faudra alors afficher un menu qui permettra à l'utilisateur de choisir ce qu'il veut effectuer.

2.3 Structure de données

Voici la structure de données qui sera utilisée dans le projet.

```
1 typedef struct ListePers {
2     struct Pers *pers;
3     struct ListePers *suivant;
4 } ListePers;
5
6 typedef struct Date {
7     int jour, mois, annee;
8 } Date;
9
10 typedef struct Pers {
11     char nom[50];
12     char prenom[50];
13     Date naissance;
14     Date deces;
15     struct Pers *pere;
16     struct Pers *mere;
17     struct Pers *conjoint;
18     struct ListePers *enfants;
19     int vivant;
20 } Pers;
```

On utilisera la structure Pers, dont les champs sont les informations sur la personne (nom, prénom, date de naissance etc) mais aussi des pointeurs sur Pers pour la mère et le père, et un pointeur sur une liste de personne pour représenter les enfants. La structure ListePers permet de créer une liste chaînée de personnes, essentielles pour créer les arbres et les manipuler.

2.4 Programme Annexe

Au vu des complications et des problèmes de compilation avec notre programme principal, nous avons choisi de rendre notre première implémentation afin d'avoir des résultats, voici la structure que nous avons rédigé au départ (et celle qui nous donnera des résultats). Voici la structure Personne que nous avons implémenté lors des premières étapes de notre projet et qui compile, elle contient le nom de la Personne, son age, son conjoint, son enfant et son parent. Nous utiliserons les pointeurs pour le conjoint, enfant et parent.

Personne

- nom : Chaine
- age : Entier
- conjoint : Personne
- enfant : Personne
- parent : Personne

3 Implémentation en C du projet principal:

L'implémentation a été effectuée sur CodeBlocks (Windows). Nous avons organisé notre projet en différents fichiers .h et .c, afin de mieux organiser notre travail mais aussi rendre plus accessible la lecture des fonctions. Ils sont disponible dans le dossier arbre.

3.1 Getters et Setters

Accéder au contenu des arbres et pouvoir les modifier.

```
char* get_nom_pers(Pers *pers);
char* get_prenom_pers(Pers *pers);
int get_jNaissance_pers(Pers *pers);
int get_mNaissance_pers(Pers *pers);
int get_aNaissance_pers(Pers *pers);
Date get_dateNaissance_pers(Pers *pers);
int get_jDeces_pers(Pers *pers);
int get_mDeces_pers(Pers *pers);
int get_aDeces_pers(Pers *pers);
int get_jour_date(Date date);
int get_mois_date(Date date);
int get_annee_date(Date date);
Pers* get_pere_pers(Pers *pers);
Pers* get_mere_pers(Pers *pers);
int get_nbEnfants_pers(Pers *pers);
Pers* get_iemeEnfant_pers(Pers *pers, int i);
ListePers* get_enfants_pers(Pers *pers);
int get_vivant_pers(Pers *pers);
Pers* get_pers_liste(ListePers *arbre);
ListePers* get_suiv_liste(ListePers *arbre);
int get_n_pers(Pers *pers);
int get_np_pers(Pers *pers);
int get_nm_pers(Pers *pers);
int get_nc_pers(Pers *pers);

void set_nom_pers(Pers *pers, const char *nom);
void set_prenom_pers(Pers *pers, const char *prenom);
void set_jNaissance_pers(Pers *pers, int jour);
void set_mNaissance_pers(Pers *pers, int mois);
void set_aNaissance_pers(Pers *pers, int annee);
void set_dateNaissance_pers(Pers *pers, Date date);
void set_jDeces_pers(Pers *pers, int jour);
void set_mDeces_pers(Pers *pers, int mois);
void set_aDeces_pers(Pers *pers, int annee);
void set_pere_pers(Pers *pers, Pers *pere);
void set_mere_pers(Pers *pers, Pers *mere);
void set_nbEnfants_pers(Pers *pers, int nbEnfants);
void set_iemeEnfant_pers(Pers *pers, int i, Pers *enfant);
void set_enfants_pers(Pers *pers, ListePers *enfants);
```

```

void set_vivant_pers(Pers *pers, int vivant);
void set_pers_liste(ListePers *arbre, Pers *pers);
void set_suiv_liste(ListePers *arbre, ListePers *suivant);
void set_n_pers(Pers *pers, int n);
void set_np_pers(Pers *pers, int np);
void set_nm_pers(Pers *pers, int nm);
void set_nc_pers(Pers *pers, int nc);

ListePers* fix_maillon_liste(ListePers* arbre);
ListePers* fix_pers_liste_creer(ListePers* arbre, Pers* pers);
ListePers* sup_maillon_liste(ListePers* arbre, Pers* intru);
ListePers* fusion_listes_sansDouble(ListePers* l1, ListePers* l2);

Pers* personneVide();
ListePers* listeVide();

```

3.2 Trouver une Personne

Ces fonctions permettent de trouver une personne.

```

int get_nbJours_mois(int m);
int estBisextile(int a);
int estUneDate(Date date);
Date demanderDate();
Pers* trouverPers(ListePers* arbre);

```

3.3 Lien entre personnes

Permet de déterminer le lien entre deux personnes saisies.

```

char* determinerLien(ListePers* arbre, Pers* pers1, Pers* pers2);

```

3.4 Manipulation fichiers

Ces fonctions permettent de charger et sauvegarder un arbre dans un fichier texte.

```

char* saisirNomFichier();
ListePers* numero(ListePers* arbre);
void sauvegarderArbreFichier(ListePers* arbre, char* nomFich);
Pers* retrouverPersavecNumero(ListePers* arbre, int n);
ListePers* retrouverLiens(ListePers* arbre);
ListePers* chargerArbre(char* nomFich);

```

3.5 Modification d'une personne

Ces fonctions permettent de modifier les attributs d'une personne mais aussi les personnes à qui elle est liée.

```

void modifierNom(Pers *pers);
void modifierPrenom(Pers *pers);

```

```

void modifierNaissance(Pers *pers);
void modifierDeces(Pers *pers);
void modifierVivant(Pers *pers);
void ajouterLienParentReci(Pers *parent, Pers *enfant, char *mpt);
void supprimerLienParentReci(Pers *parent, Pers *enfant);
void ajouterEnfant(ListePers *arbre, Pers *pers);
void ajouterPere(ListePers *arbre, Pers *pers);
void ajouterMere(ListePers *arbre, Pers *pers);
void ajouterConj(ListePers *arbre, Pers *pers);
void supprimerEnfant(Pers *pers);
void supprimerPere(Pers *pers);
void supprimerMere(Pers *pers);
void supprimerConj(Pers *pers);
void modifierPere(ListePers *arbre, Pers *pers);
void modifierMere(ListePers *arbre, Pers *pers);
void modifierEnfants(ListePers *arbre, Pers *pers);
void modifierConj(ListePers *arbre, Pers *pers);
int coherenceDates(Pers *parent, Pers *enfant, char *mpt);

```

3.6 Gestion d'un arbre

Ces fonctions permettent de modifier les "branches" de l'arbre, donc de supprimer la filiation entre les personnes, ou encore vider l'arbre.

```

Pers* creerPers();
ListePers* ajouterPersonne(ListePers* arbre, Pers* nouvellePersonne);
void supprimerFiliationPers(Pers* pers);
ListePers* supprimerPers(ListePers* arbre);
ListePers* viderArbre(ListePers* arbre);

```

3.7 Consulter les éléments d'un arbre

Ces fonctions permettent de consulter des liens entre les personnes.

```

ListePers* trouverNeveuNiece(ListePers* arbre, Pers* pers);
ListePers* trouverConjoint(ListePers* arbre, Pers* pers);
ListePers* trouverCousin(ListePers* arbre, Pers* pers);
ListePers* trouverOncleTante(ListePers* arbre, Pers* pers);
ListePers* trouverPetitsEnfants(ListePers* arbre, Pers* pers);
ListePers* trouverGrandsParents(ListePers* arbre, Pers* pers);
ListePers* trouverFratrerie(ListePers* arbre, Pers* pers);
ListePers* trouverMere(ListePers* arbre, Pers* pers);
ListePers* trouverPere(ListePers* arbre, Pers* pers);
void afficherPers(Pers* pers);
void consulterMembre(ListePers* arbre);
void afficherLien(ListePers* arbre, Pers* pers1, Pers* pers2);
void consulterLien(ListePers* arbre);
void afficherTLM(ListePers* arbre);
void afficherEnfantsRecurs(ListePers* arbre, Pers* pers);
void afficherDescendance(ListePers* arbre);

```

```
void afficherParentRekurs(ListePers* arbre, Pers* pers);  
void afficherAscendance(ListePers* arbre);  
void affichage(ListePers* arbre);
```

3.8 Menu affiché à l'utilisateur

Ces fonctions permettent d'afficher un menu à l'utilisateur et de choisir ce qu'il souhaite faire.

```
void chargement();  
ListePers* quitterSauvegarder(ListePers* arbre);  
void sauvegarder(ListePers* arbre);  
ListePers* menuPrincipal(ListePers* arbre);  
ListePers* gerer(ListePers* arbre);  
void modifierPers(ListePers* arbre);  
void consulter(ListePers* arbre);
```

Le fichier main.c ne compile pas en raison de certaines erreurs.

4 Implémentation en C du projet annexe:

4.1 Projet annexe

Ces fichiers seront disponible dans le dossier Annexe. Nous avons organisé notre projet en 3 fichier, famille.h contenant la déclaration de la structure de donnée Personne, et les fonctions utilisées dans le main.

```
1 typedef struct Personne {
2     char nom[MAX_NAME_LENGTH];
3     int age;
4     struct Personne *conjoint;
5     struct Personne *enfant;
6     struct Personne *parent;
7 } Personne;
8
9 void ajouterPersonne(Personne **famille);
10 void marier(Personne **famille);
11 void ajouterEnfant(Personne **famille);
12 void afficherFamille(Personne **famille);
13 void sauvegarderGeneralogie(Personne **famille, const char *nomFichier);
14 void chargerGeneralogie(Personne **famille, const char *nomFichier);
15 void libererMemoire(Personne **famille);
```

Les fonctions sont implémentées dans un fichier famille.c et enfin le fichier main (mainFamille.c) contient le menu et les appels de fonctions.

4.2 Description des fonctions

ajouterPersonne(Personne **famille) : Cette fonction permet d'ajouter une nouvelle personne à la famille. Elle demande à l'utilisateur d'entrer le nom et l'âge de la personne, puis l'ajoute à la liste des membres de la famille.

marier(Personne **famille) : Cette fonction permet de marier deux personnes de la famille. Elle demande à l'utilisateur d'entrer les noms des deux personnes à marier et vérifie si elles existent dans la liste. Si oui, elle les marque comme conjoints l'une de l'autre.

ajouterEnfant(Personne **famille) : Cette fonction permet d'ajouter un enfant à deux personnes mariées dans la famille. Elle demande à l'utilisateur d'entrer les noms des parents et le nom de l'enfant, puis vérifie si les parents sont mariés et s'ils existent dans la liste. Si oui, elle ajoute l'enfant à la liste des membres de la famille.

afficherFamille(Personne **famille) : Cette fonction affiche tous les membres de la famille, ainsi que leurs conjoints et enfants le cas échéant. Elle parcourt la liste des membres de la famille et affiche leurs informations.

sauvegarderGeneralogie(Personne **famille, const char *nomFichier) : Cette fonction permet de sauvegarder la généalogie de la famille dans un fichier. Elle prend en paramètre le nom du fichier dans lequel sauvegarder les données et écrit les informations de la famille dans ce fichier.

chargerGeneralogie(Personne **famille, const char *nomFichier) : Cette fonction permet de charger une généalogie à partir d'un fichier. Elle prend en paramètre le nom du fichier à charger et lit les informations de la famille à partir de ce fichier pour les ajouter à la liste des membres de la famille.

5 Résultats projet annexe

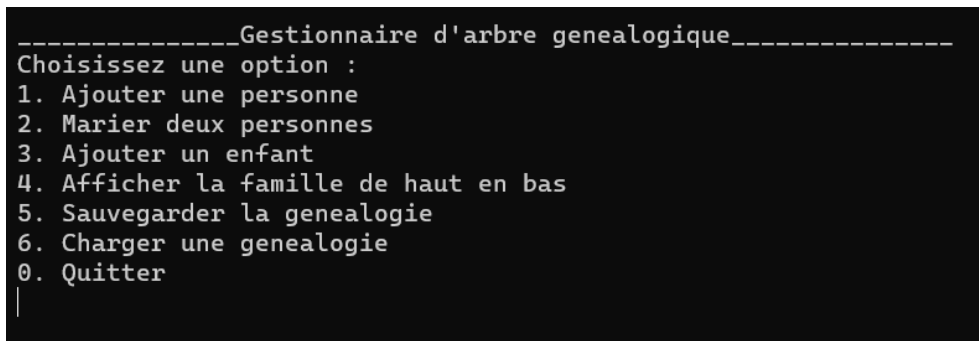
Voici les résultats que nous avons obtenus, suite à l'implémentation du projet annexe.

5.1 Affichage du Menu:

Au lancement du programme, un choix est demandé à l'utilisateur :

1. Ajouter une personne
2. Marier deux personnes
3. Ajouter un enfant
4. Afficher la famille de haut en bas
5. Sauvegarder la généalogie
6. Charger une généalogie
7. Quitter

Le menu est affiché comme ci-dessous.



```
-----Gestionnaire d'arbre genealogique-----
Choisissez une option :
1. Ajouter une personne
2. Marier deux personnes
3. Ajouter un enfant
4. Afficher la famille de haut en bas
5. Sauvegarder la genealogie
6. Charger une genealogie
0. Quitter
|
```

Figure 3: Menu affiché à l'utilisateur

L'utilisateur n'a plus qu'à choisir ce qu'il souhaite effectuer comme opération, en saisissant un chiffre et en appuyant sur Entrée.

5.2 Choix 1: Ajouter une personne

Lorsque l'utilisateur choisit la 1, il va pouvoir créer une personne.

```

-----Gestionnaire d'arbre genealogique-----
Choisissez une option :
1. Ajouter une personne
2. Marier deux personnes
3. Ajouter un enfant
4. Afficher la famille de haut en bas
5. Sauvegarder la genealogie
6. Charger une genealogie
0. Quitter
1
Entrez le nom de famille de la personne :
Dupont
Entrez le prenom de la personne :
Jean
Entrez l'age de la personne :
30
Voulez-vous declarer Dupont Jean comme chef de famille ? (o/n)
o
Dupont est maintenant le chef de famille
Appuyez sur Entree pour continuer...

```

Figure 4: Création d'une personne: Jean Dupont

L'utilisateur devra alors saisir le nom et l'âge de la personne, et il pourra aussi décider si cette personne est chef de famille (père ou mère). Cela sera important pour la phase de mariage.

Ici nous avons créé M. Jean Dupont âgé de 30 ans.

Ajoutons une nouvelle personne.

```

-----Gestionnaire d'arbre genealogique-----
Choisissez une option :
1. Ajouter une personne
2. Marier deux personnes
3. Ajouter un enfant
4. Afficher la famille de haut en bas
6. Sauvegarder la genealogie
7. Charger une genealogie
0. Quitter
1
Entrez le nom de famille de la personne :
Marie
Entrez le prenom de la personne :
Jeanne
Entrez l'age de la personne :
30
Voulez-vous declarer Marie Jeanne comme chef de famille ? (o/n)
n
Appuyez sur Entree pour continuer...|

```

Figure 5: Création d'une deuxième personne: Jeanne Marie

Ici nous avons créé Mme Jeanne Marie âgée elle aussi de 30 ans.

5.3 Choix 2: Marier deux personnes

A présent, nous avons deux personnes créées, nous pouvons donc les marier.

```

-----Gestionnaire d'arbre genealogique-----
Choisissez une option :
1. Ajouter une personne
2. Marier deux personnes
3. Ajouter un enfant
4. Afficher la famille de haut en bas
6. Sauvegarder la genealogie
7. Charger une genealogie
0. Quitter
2
Entrez le prenom de la premiere personne :
Jean
Entrez le nom de famille de la premiere personne :
Dupont
Entrez le prenom de la deuxieme personne :
Jeanne
Entrez le nom de famille de la deuxieme personne :
Marie
Jean Dupont et Jeanne Marie sont maintenant maries
Appuyez sur Entree pour continuer...|

```

Figure 6: Mariage entre deux personnes

Nous venons de marier Jean Dupont et Jeanne Marie. A noter que la fonction prend en compte l'âge, le célibat et empêche le mariage entre frères et soeurs en vérifiant que les personnes n'ont pas le même chef de famille (donc le même père).

5.4 Choix 3: Ajouter un enfant

Une fois que nos deux personnes sont mariées, nous pouvons leur ajouter un enfant.

```

-----Gestionnaire d'arbre genealogique-----
Choisissez une option :
1. Ajouter une personne
2. Marier deux personnes
3. Ajouter un enfant
4. Afficher la famille de haut en bas
6. Sauvegarder la genealogie
7. Charger une genealogie
0. Quitter
3
Entrez le prenom du pere :
Jean
Entrez le nom du pere :
Dupont
Entrez le prenom de la mere :
Jeanne
Entrez le nom de la mere :
Marie
Entrez le prenom de l'enfant :
Francois
Entrez l'age de l'enfant :
10
Francois est l'enfant de Jean Dupont et Jeanne Marie
Appuyez sur Entree pour continuer...|

```

Figure 7: Ajout d'un enfant

Encore une fois, des tests sont effectués pour vérifier que les personnes sont bien mariées afin qu'elles puissent avoir un enfant.

5.5 Choix 4: Afficher la famille

Nous avons alors une petite famille, que nous pouvons dès à présent afficher.

```
-----Gestionnaire d'arbre genealogique-----
Choisissez une option :
1. Ajouter une personne
2. Marier deux personnes
3. Ajouter un enfant
4. Afficher la famille de haut en bas
5. Sauvegarder la genealogie
6. Charger une genealogie
0. Quitter
4
Entrez le prenom de la personne :
Jean
Entrez le nom de famille de la personne :
Dupont
Nom: Dupont, Prenom: Jean, Age: 30
Conjoint: Jeanne Marie, Age: 30
Enfants:
- Francois Dupont, Age: 10
Appuyez sur Entree pour continuer...
```

Figure 8: Affichage de la famille

5.6 Sauvegarde

Cet affichage peut aussi être sauvegardé dans un fichier texte.

```
-----Gestionnaire d'arbre genealogique-----
Choisissez une option :
1. Ajouter une personne
2. Marier deux personnes
3. Ajouter un enfant
4. Afficher la famille de haut en bas
5. Sauvegarder la genealogie
6. Charger une genealogie
0. Quitter
5
La genealogie a ete sauvegardee dans generalogie.txt
Appuyez sur Entree pour continuer...|
```

Figure 9: Famille enregistrée dans un fichier texte

6 Conclusion

Ce projet de création d'arbre généalogique a été une opportunité passionnante d'appliquer les concepts de programmation et de structures de données à un domaine concret. En utilisant le langage C, nous avons pu modéliser efficacement les relations familiales en créant une structure de données flexible et adaptée à nos besoins.

Nous avons mis en œuvre les fonctionnalités de base attendues pour la gestion d'un arbre généalogique, telles que l'ajout de personnes, la création de relations familiales, l'affichage des membres de la famille, la sauvegarde et le chargement des données depuis un fichier, ainsi que la libération de la mémoire.

Au cours de ce projet, nous avons rencontré des difficultés de manipulation des structures et des pointeurs, toutefois nous avons pu nous rendre compte de l'importance de la phase de conception lors d'un projet, car c'est lors de cette phase que les structures mises en places doivent être correctement définies et prévoir des erreurs qui pourraient avoir lieu.

En fin de compte, ce projet nous a permis de développer nos compétences en programmation en C et de mieux comprendre la complexité de la modélisation des relations familiales. Il constitue une base solide pour d'éventuelles améliorations et extensions futures, telles que l'ajout de fonctionnalités avancées ou l'optimisation des performances. Nous continuerons de travailler sur ce projet afin de le finaliser et obtenir des résultats.

En conclusion, ce projet a été une expérience enrichissante qui nous a permis de mettre en pratique nos connaissances en programmation et de les appliquer à un contexte concret, tout en nous confrontant à des défis stimulants.