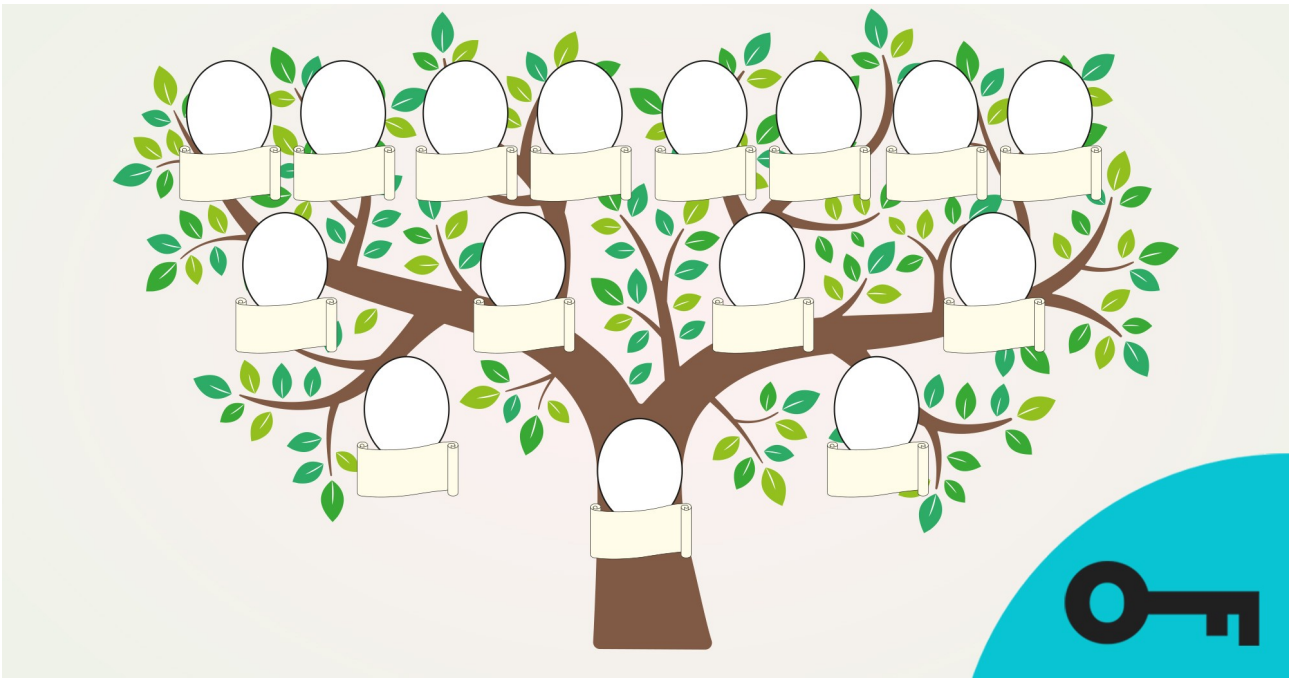
Rapport du Projet 1 : Arbre Généalogique

Noms : RAVALOSON Nomanina
BEN-AHMED-DAHO Mohamed

Equipe E707



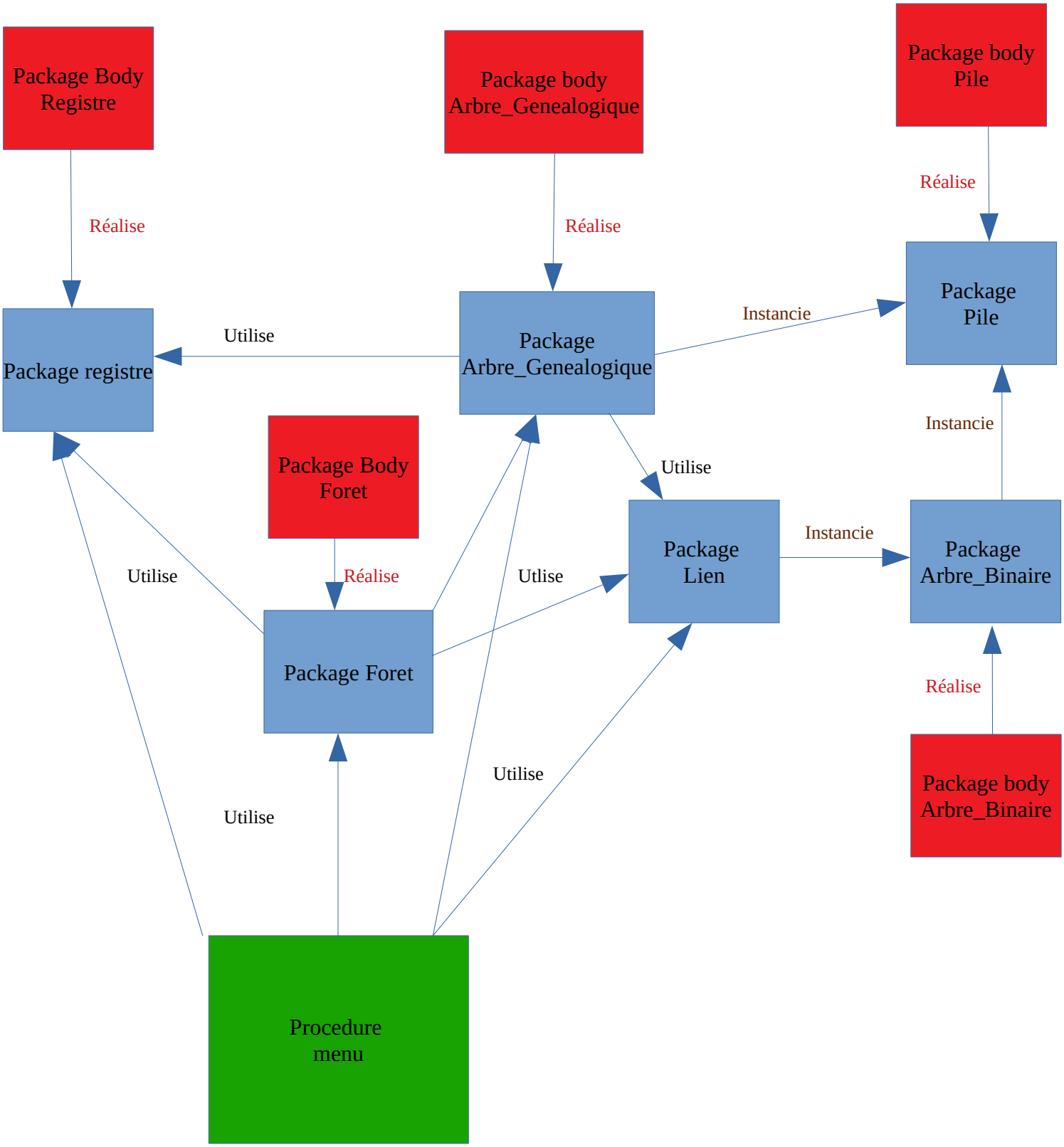
L'objectif de ce projet est de concevoir un programme de gestion d'arbre généalogique contenant une interface utilisateur conviviale et simplifiée. En plus de cet arbre, un registre est utilisé pour stocker les informations des différents individus de l'arbre. Ce rapport de projet contient l'architecture des modules implantés ainsi qu'une présentation de ceux-ci, la démarche adoptée pour la réalisation des modules, un bilan technique ainsi que 2 bilans personnels sur le projet.

Introduction :

Une lignée peut être représentée de différentes manières. On s'intéresse ici à la représenter par un arbre binaire.
Cet arbre démarre d'un nœud et est lié à deux autres nœuds : un père et une mère. Ces deux nouveaux individus sont chacun eux-même liés à un père et une mère. Cette strucutre permet de répresenter efficacement un arbre généalogique.
Chaque individu est identifié par un nombre entier unique appelé clé. Cette clé permet de différencier un individu parmi d'autres. C'est aussi grâce à cette clé que l'on peut accéder aux informations d'un individu dans le registre. Plusieurs modules ont été crées dans le but d 'implanter toutes ces fonctionnalités. Le programme final nommé Menu est l'interface de gestion de l'arbre généalogique.

PLAN :

ARCHITECTURE DES MODULES



CHOIX DE CONCEPTIONS

Nous utilisons des exceptions dans une bonne partie des programmes. Ci-dessous sont répertoriées les choix de conceptions ne concernant qu'un seul module particulier à la fois.

Arbre Binaire et lien:

Le module est très classique. On peut néanmoins noter que lors de l'écriture de ce module, nous avons décider pour l'opération de suppression d'un nœud de supprimer toutes les clés du sous-arbre qui commence à ce nœud. Ainsi, si on supprime un individu de l'arbre généalogique, tous ses ancêtres seront aussi supprimés. On utiliser aussi les piles pour certaines opérations. Le module lien instancie le module arbre binaire et donnant le type entier à la Clé.

Registre :

Nous avons opté pour une liste chaînée de listes chaînées. La liste principale est une liste chaînée de cellules contenant une clé représentée par une entier, une liste chaînée d'informations et le pointeur vers la cellule suivante. La liste chaînée d'informations, à la différence de l'enregistrements, permet plus de flexibilité sur la nature des informations que l'on souhaite rentrer. On peut en effet rentrer le nombre d'informations que l'on veut.

Arbre Généalogique :

Certaines opérations nécessitent de modifier le registre. De ce fait, le module Arbre_Généalogique utilise les modules lien et Registre pour fonctionner. On utilise aussi les piles dans certaines opérations de l'arbre. Pour certaines procédures qui demandaient de renvoyer un ensemble d'individu qui valident certains critères, on a choisi pour d'implémanter une structure de liste chaînée de clé.

Forêt :

Pour implanter la notion de forêt, nous avons décidé d'utiliser le registre poru stocker les relations entre les individus ce qui est efficace étant donné la nature de notre registre. La forêt en tant que telle est simplement une liste chaînée d'arbre généalogique.

Présentation des principaux algorithmes et types de données

Intéressons-nous tout d'abord aux types de données définis :

Type T_Arbre_Binaire : C'est le Type utilisé pour représenter l'arbre binaire. C'est un pointeur vers un type T_Cellule_Arbre qui est un enregistrement de la clé de type générique T_Clé et de deux pointeurs gauche et droite de type T_Arbre_Binaire. C'est un type privé.

Type T_Registre : C'est le type utilisé pour représenter le registre. C'est un pointeur vers un type T_Cellule_Registre qui est un enregistrement d'une clé de type entier, d'un pointeur Information de type T_Info et d'un pointeur sur la cellule suivante. C'est un type très privé.

Type T_Info : C'est le type utilisé pour représenter la liste chaînée d'informations pour un individu. C'est un pointeur vers un Type T_Cellule_Info qui est un enregistrement d'une information de type NSTR et et d'un pointeur vers la cellule suivante . C'est un type très privé.

Type NSTR (pour New String): C'est le type utilisé pour représenter les informations. C'est un sous-type de string de taille limitée à 45. Cette limitation de taille permet à l'ordinateur d'allouer une quantité finie de mémoire pour stocker l'information. Ce type peut ainsi être utilisé dans une liste chaînée.

Type T_Liste : C'est le type utilisé pour représenter des listes chaînées de clé. C'est donc un pointeur vers une cellule de type cellule qui est un enregistrement d'une clé de type entier et d'un pointeur vers la cellule suivante. C'est un type très privé.

Type T_Arbre_Gen : C'est le type utilisé pour représenter l'arbre généalogique. C'est un sous type d'arbre binaire ce qui permet d'avoir les mêmes caractéristiques que le type T_Arbre_Binaire.

Type T_Foret : C'est le typpe utilisé pour représenter la forêt. C'est une liste chaînée de T_Arbre_Gen.

Présentation des principaux algorithmes et types de données

Passons maintenant à la présentation des principaux programmes :

procedure **Initialiser_Arbre_Gen**(A : out T_Arbre_Gen;R: out T_Registre;Cle: in Integer)
Initialise une arbre généalogique ainsi qu'un registre.

procedure **Ajouter_Arbre_Gen**(A: in out T_Arbre_Gen;R: in out T_Registre; Cle_parent: in Integer ;Cle_enfant: in Integer; choix: in Boolean)
Ajoute la clé parent selon le choix de l'utilisateur à la clé enfant et ajoute la clé parent au registre.

procedure **Supprimer_Noeud_Gen**(A: in out T_Arbre_Gen; Cle: in Integer; R: in out T_registre)
Supprime le nœud ainsi que tous les ancêtres du nœud.

procedure **Afficher_Arbre_Gen**(A : in T_arbre_gen ;Cle : in Integer)
Afficher l'arbre généalogique.

procedure **Ajouter_R_info**(R: in out T_Registre;Cle: in Integer)
Ajoute une information entrée au clavier pour la clé donnée dans le registre.

function **Obtenir_Nom**(R:in T_Registre; Cle:in Integer) return NSTR
Cherche le nom d'une clé dans le registre et le renvoie.

procedure **Supprimer_R_information**(R:in out T_Registre; Cle:in Integer)
Demande à l'utilisateur quel information supprimer et la supprime.

procedure **Afficher_R**(R:in T_Registre)
Afficher toutes les informations de toutes les clés du registre.

Démarche de test du programme

Pour tester le programme, nous avons tout d'abord testé séparément les différents modules (arbre_binaire, registre,arbre_généalogique et forêt) dans différents programmes de tests c'est pourquoi nous avons 4 programmes de tests. Chacun de ces programmes de tests s'assure que les fonctions et les procédures implantées fonctionnent de la façon attendue. Nous n'avons toutefois pas implantés la notion de gestion d'exception dans les programmes de tests.

Une fois le programme Menu réalisé, nous avons testés différents scénarios générant ou non des exceptions afin de s'assurer que la gestion de celles-ci était bien réalisée.

Difficultées rencontrés/Solutions Adoptées

Difficulté 1 : Comment ajouter une information dans le registre ?

Étant donné que nous avons opté pour une information de type NSTR, l'information entrée pouvait par exemple ne pas utiliser les 45 caractères ce qui générait des erreus. Il fallait donc récupérer la longueur de la chaîne de caractère entrée et initialiser les caractères d'indice supérieurs à la longueur à ' ' c'est-à-dire un espace.

Difficulté 2 : Comment implanter la fonction Homonyme ?

La fonction Obtenir_nom renvoie le nom de la personne dans le cas où le nom est renseigné. Dans le cas contraire elle renvoie une chaîne de caractère indiquant que le nom n'est pas renseigné. Cela posait un problème dans la fonction Homonyme qui compare 2 à 2 les noms d'individus grâce à la fonction Obtenir_nom. Si les 2 individus comparés n'ont pas de noms dans le registre, la fonction Obtenir_nom renvoie le même résultat pour les 2 individus ce qui faisait que la fonction Homonyme considérait que ces 2 individus avaient le même nom.

Pour résoudre ce problème, il fallait ajouter une condition losque l'on compare les 2 individus : si le résultat de Obtenir_Nom de l'un est "Le nom de cet individu n'est pas renseigné " alors la comparaison n'a pas lieu et on passe à la comparaison suivante .

Difficulté 3 : Comment implanter Obtenir_nb_ancetres_gen ?

Au départ, on n'avait pas de notion de génération. L'implantation a donc été compliquée en version itérative. On est donc retourné en récursif en définissant une fonction qui appelle une procédure récursive.

Organisation du groupe

Travail effectué par Nomanina :

Implantation et spécification de :
-Arbre_Binaire
-La première moitié de Arbre_Généalogique
-Piles
-Getters
-Menu (version sans forêt)
-Forêt

Travail effectué par Mohamed :

Implantation et spécification de :
-Registre
-La seconde moitié de Arbre_Généalogique
-Types et opérations associées aux listes chainées
-Menu (version avec forêt)

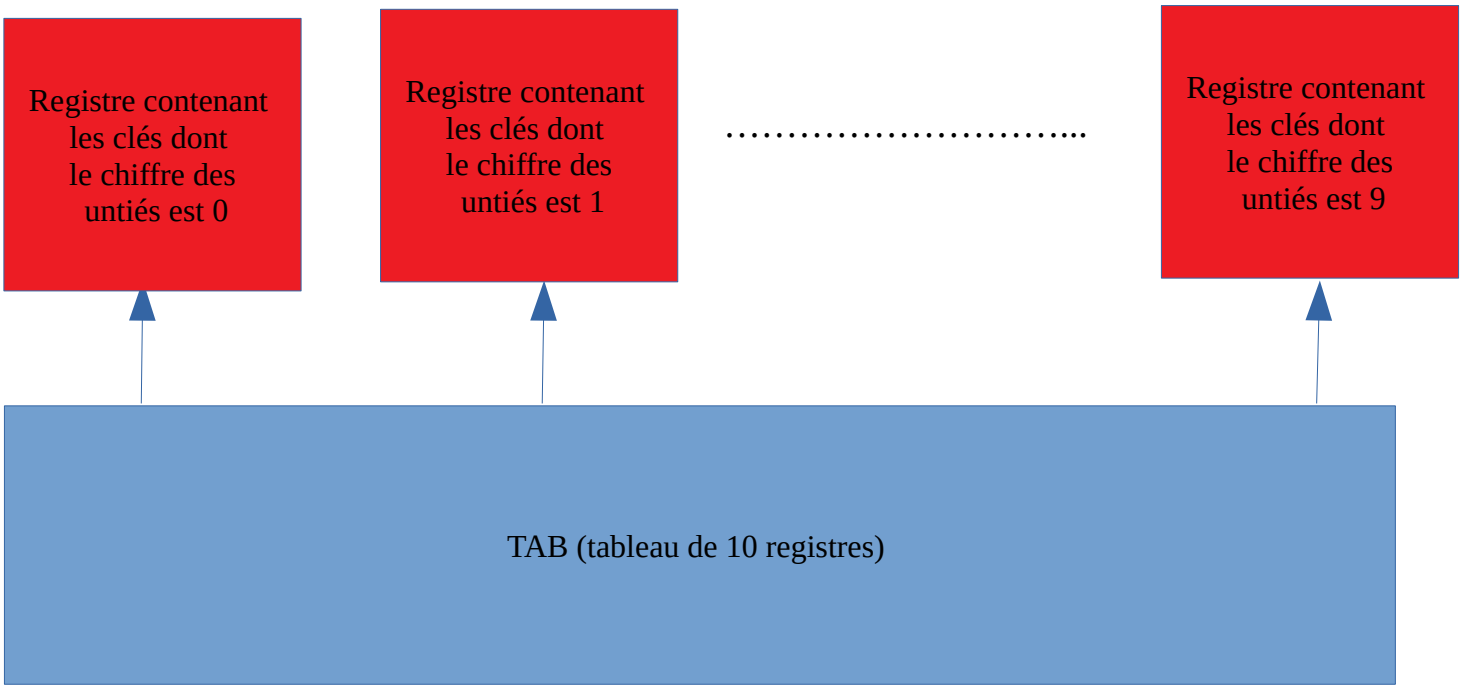
Bilan Technique

Le menu fonctionne bien et il n'y a pas de problèmes à ce jour.La notion de forêt a aussi été implantée. L'interface est satisfaisante. La totalité des fonctions demandées a bien été implantée.

<u>Perspectives d'évolution</u> :

La structure du registre semble être réactive pour le nombre d'individu testés. On pourrait améliorer le temps de parcours de la façon suivante :
On définit un type **TAB** qui est un tableau de 10 registres définis comme précedemment. Chaque registre contient uniquement les informations des clés dont le chiffre des unités est égal à la position du registre dans le **TAB**. Lorsque l'on recherche une clé, il suffit de regarder le chiffre des unités et se déplacer à la case du tableau d'indice égal à ce chiffre puis parcourir normalement le registre associé à ce chiffre des unités. On divise en moyenne ainsi par 10 le temps de parcours puisque dans le pire des cas on parcourt seulement le nombre total de clés divisé par 10.

On peut biensur réitérer cette méthode en faisant pointer chaque élément de **TAB** vers un autre **TAB** pour les chiffres des dizaines ce qui permettrait de diviser en moyenne par 100 le temps de parcours.



Bilans personnels :

J’ai trouvé ce projet très intéressant car il m’a permis de synthétiser toutes les connaissances accumulées en PIM. La partie la plus intéressante fût pour moi l’implantation du registre. J’ai apprécié la structure liste chaînée de listes chaînées ainsi que le fait que le fait que l’on peut rentrer n’importe quelle type d’informations. J’ai passé environ 7h sur le registre et 3h sur la seconde partie de arbre généalogique. Le rapport m’a pris 3h à faire. Grâce à ce projet, j’ai appris que l’on pouvait restreindre la taille d’une chaîne de caractères et la fixer.

Mohamed

L’élaboration du menu a été longue mais satisfaisante. J’ai bien aimé la gestion d’exceptions car c’était pour moi un challenge personnel. Aussi, l’implantation principalement en itératif du module Arbre_Binaire m’a plu car j’ai trouvé cela plutôt original. J’ai passé 2h sur l’implantation de ce module et 4h sur la première partie de arbre généalogique et 3h sur le menu. Ce projet m’a enseigné comment consolider mes connaissances en algorithmique en particulier la généricité.

Nomanina