Rapport du Projet : Programmation impérative

1 - Résumé :

Dans ce rapport, nous allons discuter de manière détaillée et précise toutes les procédures par lesquelles nous sommes passé pour réaliser notre projet. À travers ce rapport, nous cherchons à vous approcher de notre cheminement que nous avons suivi pendant la réalisation progressive de notre travail. Tout d'abord, nous allons vous initier à la problématique du sujet puis nous allons vous parler des différents choix que nous avons procédé pour traiter le problèmes aussi que les obstacles que nous avons rencontrer, ensuite nous allons vous parler d'un bilan technique qui décrit chronologiquement l'évolution de notre travail pour arriver finalement à un bilan personnel et individuel contenant nos acquis tirés à partir de notre propre réalisation du projet .

2 - Introduction :

Le probleme posé touche à une problématique qui demeure cruciale dans notre vie, c'est s'identifier et se mettre en quete d'une recherche identitaire qui va nous permettre de connaître nos ancetres . Dans cette direction, le sujet consiste à Identifier des individus par leurs informations personnelles en les liant avec leurs ancetres à l'aide des arbres généalogiques. Le problème mis à notre disposition propose alors de manipuler de manière harmonique les arbres généalogiques tout en tenant compte d'un registre d'état civil qui va contenir à son tour les informations des personnes identifiées dans l'arbre généalogique. Ce registre peut contenir plusieurs informations, dans ce sujet on se limitera aux noms, prenoms , date de naissance et lieu de naissance .

Remarque : Nous avons traité les deux parties du projet .

3 - L'architecture de l'application en mobile :

Pour réaliser ce travail, nous avons procédé par la méthode de raffinages qui nous a souligné les grands axes que nous devons suivre d'une manière concise tout en prévoyant une possibilité de changement. A savoir, nous nous proposons d'élaborer

un module générique ‘arbre_binaire’ qui va contenir toutes les opérations élémentaires qui manipulent les arbres (Création d’un arbre , Ajout d’un fils gauche et d’un fils droit , Calcul de la taille, Suppression d’un nœud …). Ensuite, on s’est proposé de créer un module ‘registre’ de type table de hashage qui va contenir des dictionnaires d’état civil, et finalement , on a crée un module générique ‘Arbre_genealogique’ qui se pose sur les deux modules précèdants, ce dernier est celui qui fait la corrélation entre la structure d’un registre et la structure d’un arbre genealogique.

4 - Principaux choix réalisés :

- Le module Registre :

On a choisi une structure abstraite de données sous forme d’un tableau de hashage. Pourquoi un tableau de hashage?

- Le tableau ne serait pas un solution efficace à ce sujet car le tableau est définie par une capacité, chose que nous ne pouvons pas connaître, ce qui fait qu’on va laisser des cases mémoire vide sans interet.
- A première vue , la liste chainée apparaît une solution efficace car elle n’est pas définie à partir d’une capacité toutefois, la recherche en ce type de données ne serait pas assez simple et rapide comme demandé car à chaque fois il faut parcourir plusieurs élèments depuis le début pour arriver à l’individu cible.
- La solution efficace est l’utilisation d’un tableau de hashage qui est un type abstrait de données permettant l’acces rapide aux élèments sans laisser des cases mémoires vides.

- Le module Arbre_Binaire :

Un module générique qui permet la manipulation des arbre binaires à l’aide de plusieurs sous-programmes .

- Le module Arbre_Genealogique :

Un module générique qui met en relation les deux autres modules pour permettre à l’utilisateur de manipuler les individus de manière efficace et avec la capacité qu’il veut.

- Le Module Arbre_Foret :

Un module générique qui permet de manipuler un arbre plus compliqué que l'arbre binaire vu qu'il contient aussi les compagnons et les conjoints d'un individu donné .

5 - Principaux algorithmes et types de données :

1 - Type de données :

Partie I :

- Le type T_Abr est un pointeur sur une cellule de type enregistrement T_Noeud contenant la clé de type T_Cle, le sous_arbre_droit de type T_Abr et le sous_arbre_gauche de type T_Abr .
- Le type T_Cle est un type générique qui doit contenir l'identifiant d'une personne.
- Le Type T_Registre est un enregistrement contenant un tableau de type T_Tab_Registre et la taille de type entier.
- Le type T_Tab_Registre est tableau des T_Cellule, chaque T_Cellule est un enregistrement contenant un identifiant de type Integer et un dictionnaire de type T_Dictionnaire.
- Le type T_Dictionnaire est un enregistrement contenant le prénom de type String , le nom de type String , la date de naissante de type T_Date et le lieu de naissance de type String.
- Le type T_Identifiant est un type utilisé dans le module arbre_genealogique qui instance le module arbre_binaire par T_Identifiant (T_Identifiant => T_Cle)
- Le type T_Ensemble_Identifiant est un pointeur sur une cellule T_Cellule_Identifiant contenant l'identifiant de type T_Identifiant et suivant de type T_Ensemble_Identifiant.
- Le type T_Abr et T_Registre sont privés et limités afin d'interdire à l'utilisateur de les manipuler soit par modification directe , affectation ou comparaison, ce ne serait que par les services offerts par notre interface. C'est le principe de l'encapsulation.

- Le type T_Cle et T_Identifiant sont des types privés en vue d'interdire à l'utilisateur de les manipuler aléatoirement.

- La généricité utilisée pour les types T_Cle et T_Identifiant permet à l'utilisateur de choisir le type dont il veut définir les identifiants de ses individus. Pour les tests , on adoptera le type Integer des entiers .

Partie II :

- Le Type T_Abr_Foret est un type utilisé dans le module arbre_foret , il est un pointeur sur une cellule T_Cellule contenant la clé de type Integer , le Père de type T_Abr_Foret , la mère de type T_Abr_Foret , l'époux (se) de type T_Abr_Foret et contient finalement un tableau des compagnons de type T_Tableau.
- Le type T_Tableau est un enregistrement contenant un tableau de type T_Tab et sa taille.
- Le Type T_Tab est un tableau contenant des élèments de type T_Abr_Foret.

2- Algorithmes principaux :

Partie I:

1. **Obtenir les individus qui ont un seul parent.**

Ces fonctions contiennent comme variable un tableau T de type T_Ensemble_Identifiant et une procédure Ajouter qui réalise un parcours de toute l'arbre en vérifiant si un nœud a uniquement un seul parent, si c'est le cas cette procédure ajoute l'identifiant de cet personne dans le tableau T. On fait l'appel de la procédure Ajouter dans le corps de la fonction Aucun Parent et on retourne le tableau T .

De meme pour les fonctions qui determinent si un individu a deux parents / Aucun parent .

2 . Verifier si deux individus ont en minimum un ancetre commun .

On crée deux élèments T1 et T2 de types T_Ensemble_Identifiant et on les affecte respectivement l'ensemble des ancetres des deux individus. On crée un

élèment intermédiaire T de type T_Ensemble_Individu et fait le parcours de T1 et T2 si on trouve un élèment commun entre ces deux derniers on l'ajoute a T.
En fin de compte , on aura tous les ancetres communs aux deux individus (il se peut qu'ils n'auront aucun ancetre commun) et retourne la valeur booléane de l'expression (T!= Null).

3 . Afficher un arbre binaire.

Pour afficher un arbre binaire on va utiliser une procedure générique dans le module 'Arbre_Binaire' : Afficher_Arbre qui va afficher tout l'arbre à l'aide d'une autre procédure Afficher_Element qui a pour but l'affichage des élèments de l'arbre selon leur type .

Partie II :

4 . Afficher les demis frères/sœurs de la part du père :

On fixe un individu de l'arbre et on fait un parcours dans les arbres associés à ses compagnons, si on trouve , pour un compagnon donné , le meme père que l'individu de la racine , on vérifie s'ils n'ont pas la meme mère , si c'est le cas on l'affiche sinon on n'affiche rien.
On fait de meme pour les demis frères/sœurs de la part de la mère.

6 - La démarche adopté pour tester les programmes :

Pour tester les programmes arbre_binaire.adb, arbre_genealogique.adb et registre.adb on a implanté respectivement des fichiers test_arbre_binaire.adb , test_arbre_genealogique.adb et test_registre.adb .

1 - test_arbre_binaire.adb

Ce test vérifie la pertinence et la robusstesse de chaque service offert par la spécification du module arbre_binaire. En effet , le test construit un arbre binaire définit au sein du fichier et l'affiche sur l'écran pour vérifier si la construction est correcte , puis, le test vide l'arbre construit et vérifie qu'il est vide , ensuite , il modifie et supprime la valeur de quelques identifiants , finalement le test vérifie la validité de la procédure existe que l'on va utiliser régulièrement.

2 – test_registre.adb

En premier lieu , on a utilisé un type de liste chainée pour le registre , apres avoir remarquer l'inconvénient des listes chainées au niveau des parcours on a changé le type de registre vers un tableau de hashage et donc un nouveau test pour le registre :

Ce nouveau test crée en premier lieu un dictionnaire avec un prénom , nom , date de naissance , lieu de naissance puis il ajoute ce dictionnaire à un identifiant déjà initialisé dans le registre, ainsi le registre n'est pas vide. On retourne ce dictionnaire , on le vide après et on vérifie qu'il est vraiment vide.

3 - test_arbre_genealogique.adb

Ce test a pour but de tester les procédures et les fonctions qui sont spécifiques à l'arbre généalogique. En effet , on construit un arbre binaire décrit dans le corps du fichier et on fait un parcours de l'arbre pour avoir les individus qui ont un seul/ deux / aucun parent(s) et on vérifie si les tableaux sont bien identiques à ceux déclarés par nous.

4 – test_arbre_foret.adb

Ce test a pour but de tester les procédures et le fonctions qui sont spécifiques à l'arbre foret. En effet , on vérifie la validité des procédures qui ont pout but la construction de l'arbre foret . On vérifie aussi la procédure qui affiche les demis frères de la part du père et la procédure qui affiche les demis frères de la part de la mère.

7 - Les difficultés rencontrées et les solutions adoptées :

Partie I :

- Nous avons hésité pour choisir le type abstrait de données que l'on va utiliser pour le module registre. Nous avons pensé en premier lieu à une structure de tableau, le tableau est efficace en ce qui concerne l'accés toutefois on ne peut pas savoir la taille de l'arbre donc on risque d'utiliser des cases mémoires qu'on va pas exploiter.

On a ensuite utilisé la liste chainée comme type abstrait de données pour le registre, après avoir cherché sur internet et se renseigné auprès des professeurs, on a découvert que la liste chainée reste une solution acceptable cependant elle n'était pas pratique pour la recherche et elle ne répond pas à l'exigence de la rapidité, on a ensuite déduit qu'on doit utiliser un tableau de hashage qui répond aux deux consignes, cependant , on ne maitrisait pas l'implantation de ce TAD donc on a du cherché sur ce type afin de le maitriser .

- Pour le dictionnaire qui va contenir les informations personnelles d'un individu, on a pensé à créer un module à part les modules qu'on a et qui va contenir aussi des services pour manipuler les dictionnaires, cependant, on n'aura pas besoin des procédures ou fonctions compliqués pour les manipuler on a opter alors pour définir un type T_Dictionnaire comme un enregistremment encapsulé dans le module «arbre_genealogique» . Il en est de meme pour le type T_Date.

Partie II :

- Nous avons trouvé une difficulté supreme pour choisir la structure qui va modéliser l'arbre foret, il en existe plusieurs mais la difficulté réside dans le fait de choisir la plus simple en vue de la conception et la manipulation aussi. Nous décidé finalement de stocker les compagnons hormis l'époux(se) d'un individu dans un tableau dont les élèments sont de type T_Abr_Foret . Ainsi, nos compagnons font partie de tous les proches d'un individu hormis le père , la mère , l'époux(se) .

- Il existe deux façon de décrire des demis frères/sœurs, soit qu'ils sont des demis frères/sœurs de la part du père (ils ont le meme père mais ils n'ont pas la meme mère) soit qu'ils sont des demis frères/sœurs de la part de la mère (ils ont la meme mère mais ils n'ont pas le meme père). Par conséquent , on définit 2 procédures qui affichent ces demis frères/sœurs, une qui affiche les demis frères/sœurs de la part de la mère et une autre qui affiche les demis frères/sœurs de la part du père.

8 - Organisation du groupe :

Pendant notre raffinage et notre discussion à propos des solutions que l'on va envisager on a bien partagé les roles entre moi et mon camarade , certes on a partagé les roles mais on a travaillé tout ensemble durant toute la durée associée au projet comme un seul élève qui travaille tout le sujet et non pas deux.

- Faical TOUBAL HADAOUI s'est interessé par les structures des arbres (arbre_binaire et arbre_genealogique) et tous ce qui les concernent (Spécification et implantation et tests) et aussi il s'est chargé de réaliser ce rapport.

- Ayoub AASSOU s'est interessé par le registre et le type que l'on doit associer et s'est engagé pour le spécifier , l'implanter et le tester , il s'est chargé aussi pour réaliser le manuel d'utilisateur .

- Pour la partie 2 , on a décidé de la travailler toute entiere de manière collective vue sa difficulté .

9 - Bilan Technique :

Pour résoudre la problématique du sujet nous somme passé par plusieurs étapes intéressantes que nous avons bien noté dans notre agenda qui décrit l'évolution au cours du sujet chaque fois qu'on avance. Les étapes majeures sont décrites en ce qui suit :

06/11/2019 & 08/11/2019 : Avant tout , Nous avons lu attentivement tout le sujet et nous avons décortiquer tout ce qui demande de nous comme étant des lecteurs.

12/11/2019 : Moi et mon binome, nous avons partagé les roles entre nous de manière équipartitionnée et conviviale, chacun de nous s'est engagé à travailler sur les problématiques auxquelles il se voit fort tout en les discutant avec l'autre.

À partir de 13/11/2019 jusqu'au 15/11/2019 : nous avons débuté par un raffinage de chaque module pour que les idées soient très claires et la liaison entre les

différents module soit bien établie. Ensuite, chacun de nous a commencé par la spécification du module qu'il prend en charge.

Du 16/11/2019 au 21/11/2019 : nous avons realisé les spécifications de tous les modules ainsi que leurs programmes de test.

Du 02/12/2019 jusqu'au 09/12/2019 : nous avons commencé à implanter nos modules et le programme principal tout en prenant compte de la compilation et l'éxecution valide et conviviale de chacun d'eux.

Du 10/12/2019 jusqu'au 13/12/2019 : On s'est mis à réaliser le rapport et le manuel d'utilisateur .

- Pendant la conception de ce projet, on discutait à propos de chaque service qu'on doit offrir à l'utilisateur afin de lui permettre une utilisation fluide et amicale sans aucun problème. Pour ce faire, on fréquentait assez souvent les professeurs qui nous ont aidé énormément et on se renseignait aussi auprès des élèves de la deuxième année qui possédent bien beaucoup plus d'éxpériences que nous.

Du 14/12/2019 Jusqu'au 20/12/2019 : On s'est engagé à préparer la partie 2 qui traite la notion d'une foret à la place d'un simple arbre généalogique.
On a commencé par choisir les types qui vont décrire notre structure de la foret et on s'est mis à implanter les différentes procédures et fonction pour finir par réaliser un test qui vérifie la validité des services offerts par notre structure.

10 - Bilan Personnel :

Faical TOUBALI HADAOUI :

Interet : La passion de travailler sur une problématique avec un esprit d'ingénieur.
Temps passé: 30h
Temps passé à la conception: 7h
Temps passé à la mise au point: 21h
Temps passé sur le rapport: 2h
Enseignements tirés :

- Le travail d’équipe doit etre le but principal de tout projet collectif car ça permet d’apprendre à travailler en groupe et c’est ça l’esprit créatif d’un ingénieur .

- Le projet nous a offert une opportunité pour solidifer nos connaissances en programmation en Ada et découvrir les erreurs techniques les plus capricieux qu’un programmeur en Ada peut commettre .

Ayoub AASSOU :

Interet : L’organisation du temps et de la méthode
Temps passé: 30h
Temps passé à la conception : 10h
Temps passé à la mise au point : 18h
Temps passé sur le manuel d’utilisateur: 2h
Enseignements tirés :

- Le travail sur ce projet m’a permis personnellement de gérer plus au moins mon stress et avoir de la patience devant les problématiques que je peux rencontrer.