14/01/2026

Eliott Blanchard
Reda Aouad

Compte rendu SMP
TP5

Q1 et Q4. Construction d'un type Personne

Dans un fichier type_def.h :

code :

TP5 - type_def.h

```
struct personne {
    string nom;
    string prenom;
    u_int16_t date;
    u_int8_t genre;
    personne *conjoint;
    personne *pere;
    personne *mere;
};
```

Dans un fichier utilitaires.cpp :

code :

TP5 - utilitaires.cpp

```
// créer une personne à partir des informations nécessaires
personne * generer_personne() {
    personne * nouveau = new personne;
    cout<<"entrez un prenom :"<<endl;
    cin>> nouveau->prenom;
    cout<<"entrez un nom :"<<endl;
    cin>> nouveau->nom;
    cout<<"entrez une date :"<<endl;
    cin>> nouveau->date;
    cout<<"entrez un genre 0 si fille, 1 si garcon :"<<endl;
    cin>> nouveau->genre;
    cout<<endl;
    nouveau->conjoint = nullptr;
    nouveau->pere = nullptr;
    nouveau->mere = nullptr;
    return nouveau;
}
```

Fonction : crée une nouvelle personne.

Paramètres : aucun.

Renvoie : un pointeur vers une personne créée (`personne*`)

Q2. Mariage

code :

TP5 - utilitaires.cpp

```cpp
// marie les deux personnes et modifie leur champ conjoint
void mariage(personne * personne1, personne *personne2) {
    if(mariage_possible(personne1,personne2)){
        cout<<"mariage impossible";
        return;
    }
    cout<< "mariage des 2 personnes"<<endl;
    personne1->conjoint = personne2;
    personne2->conjoint = personne1;
}
```

Fonction : lier deux personnes comme conjoints.

Paramètres : deux pointeurs vers des personnes.

Renvoie : rien (`void`).

fruit :

cas impossible :

```
mariage impossibleeliottb@eliottb-ThinkPad-X
entrez un prenom :toto
entrez un nom :a
entrez une date :1111
entrez un genre 0 si fille, 1 si garcon :1

entrez un prenom :toto
entrez un nom :a
entrez une date :1111
entrez un genre 0 si fille, 1 si garcon :1

mariage impossibleeliottb@eliottb-ThinkPad-X
```

cas possible :

```
entrez un prenom :toto
entrez un nom :A
entrez une date :1990
entrez un genre 0 si fille, 1 si garcon :1

entrez un prenom :tata
entrez un nom :B
entrez une date :1990
entrez un genre 0 si fille, 1 si garcon :0

[x] mariage possible
```

Q3. Affichage simple

code :

TP5 - utilitaires.cpp

```cpp
// Affiche une personne : montre le prénom, le nom et le téléphone du maillon reçu.
void  affichagePersonne(personne *personne1) {

    if (personne1 == nullptr) {cout<<"vide"<<endl;
    }
    else{
    cout<<"Prenom : " << personne1->prenom<< endl;
    cout<<"Nom : " << personne1->nom<< endl;
    cout<<"date de naissance :"<<personne1->date<<endl;


        if(personne1->genre){
            cout<<"Masculin"<<endl;
        }
        else {
            cout<<"Feminin"<<endl;
        }
        if (personne1->conjoint == nullptr){
        cout<<"recherche amour"<<endl;
        }
        else    {
            if  (personne1->conjoint->genre == 1)   {
                cout<<"épouse de : "<< personne1->conjoint->prenom; cout<< personne1->conjoint->nom<<endl;
            }
            else    {

                cout<<"époux de : "<< personne1->conjoint->prenom; cout<< personne1->conjoint->nom<<endl;
            }

        }
        if (personne1->pere ==nullptr)  {
            cout<<"de pere INCONNU "<<endl;
        }
        else {
            cout<<"pere : "<< personne1->pere->prenom; cout<< personne1->pere->nom<<endl;

        }
                if (personne1->mere ==nullptr)  {
            cout<<"de mere INCONNUE "<<endl;
        }
        else {
            cout<<"mere : "<< personne1->mere->prenom; cout<< personne1->mere->nom<<endl;

        }

    cout<<endl;
    }
}

```

Fonction : affiche les informations d'une personne.

Paramètres : un pointeur vers une personne.

Renvoie : rien.

fruit :

```
entrez un prenom :
eliott
entrez un nom :
blanchard
entrez une date :
2004
entrez un genre 0 si fille, 1 si garcon :
1

Prenom : eliott
Nom : blanchard
date de naissance :2004
Masculin
recherche amour
de pere INCONNU
de mere INCONNUE
```

Q5. Frère et soeur + même

code :

TP5 - utilitaires.cpp

```cpp
// vérifie si deux indivudus sont frère ou soeur
bool frere_soeur(personne *personne1,personne *personne2)   {
    if (personne1->pere == personne2->pere  && personne1->mere == personne2->mere &&
        personne1->pere != nullptr && personne2->pere != nullptr &&
    personne1->mere != nullptr && personne2->mere != nullptr) return true;
    else return false;
}

```

Fonction : vérifie un lien de fratrie et s'il ne sont pas orphelins

Paramètres : deux pointeurs de personnes.

Renvoie : un booléen (`true` ou `false`).

Q6. Ancêtres

code :

TP5 - utilitaires.cpp

```cpp
// vérifie si un individu b est un ancetre d'un individu a
bool ancetre (personne *personne_a, personne *personne_b) {
    if (personne_a == nullptr) return false;

    if((personne_a->pere == personne_b) || (personne_a->mere == personne_b)) return true;

    return ancetre(personne_a->pere, personne_b) || ancetre(personne_a->mere, personne_b);
}
```

Fonction : vérifie un lien d’ascendance (ancêtre).

Paramètres : deux pointeurs de personnes.

Renvoie : un booléen.

Q7. Générations

code :

TP5 - utilitaires.cpp

```cpp
// compte le nombre de génération d'un arbre
u_int8_t generation (personne * personne1)  {

    //cas ou la perosnne n'existe pas
    if (personne1 == nullptr) return 0;

    int hauteur_pere = 1 + generation(personne1->pere);
    int hauteur_mere = 1 + generation(personne1->mere);

    if  (hauteur_mere<hauteur_pere)   {
        return hauteur_mere;
    }
    else return hauteur_pere;

}

```

Fonction : calcule la hauteur générationnelle de l'arbre.

Paramètres : un pointeur vers une personne.

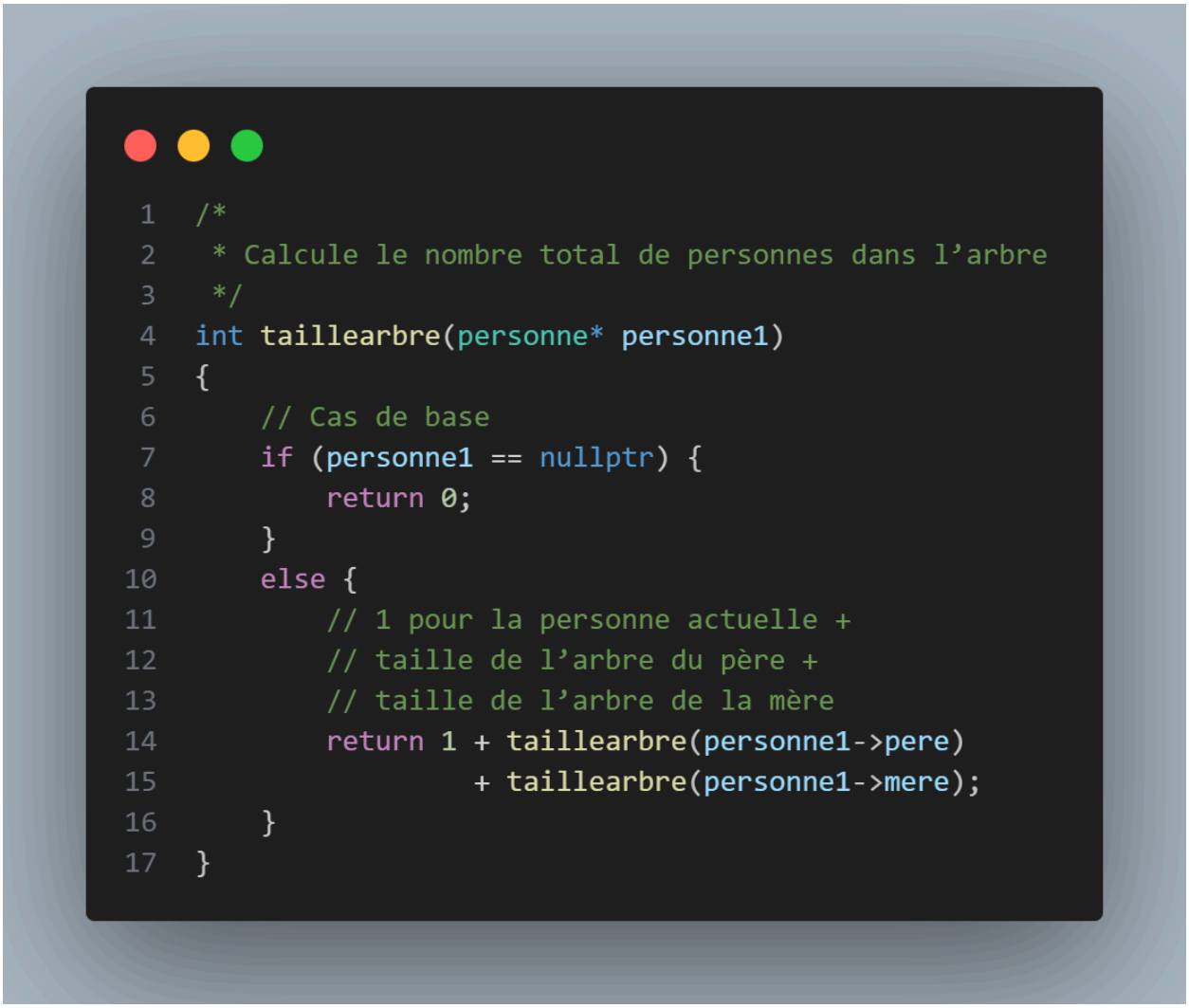
Renvoie : un entier non signé (`u_int8_t`).

fruit :

```
entrez un prenom :toto
entrez un nom :A
entrez une date :1980
entrez un genre 0 si fille, 1 si garcon :1

entrez un prenom :titi
entrez un nom :B
entrez une date :1980
entrez un genre 0 si fille, 1 si garcon :0

entrez un prenom :tata
entrez un nom :Z
entrez une date :2000
entrez un genre 0 si fille, 1 si garcon :1

[x] mariage possible
[x] assigniation d'un pere et d'une mere
nombre de géneration : 2
Prenom : toto
Nom : A
date de naissance :1980
Masculin
recherche amour
pere : titi B
mere : tata Z
```

Q8. Taille de l’arbre

code :



```
/*
 * Calcule le nombre total de personnes dans l’arbre
 */
int taillearbre(personne* personne1)
{
    // Cas de base
    if (personne1 == nullptr) {
        return 0;
    }
    else {
        // 1 pour la personne actuelle +
        // taille de l’arbre du père +
        // taille de l’arbre de la mère
        return 1 + taillearbre(personne1->pere)
                 + taillearbre(personne1->mere);
    }
}
```

Fonction : compte les membres d’un arbre familial.

Paramètres : un pointeur vers une personne racine.

Renvoie : un entier non signé (`u_int8_t`).

fruit :

```
--- MESURES DE L'ARBRE ---
Generation de Emma  : 3 (Attendu: 3)
Generation de Paul  : 1 (Attendu: 1)
Taille arbre (Emma) : 5 (Attendu: 5 )
---------------------------
```

Q9. Mariage possible

code :

TP5 - utilitaires.cpp

```cpp
// vérifie si deux personnes sont compatibles au mariage
bool mariage_possible(personne *personne_a, personne *personne_b)   {
    if (ancetre(personne_a, personne_b) ||
    ancetre(personne_b, personne_a) ||
    frere_soeur(personne_a, personne_b)) return false;
    return true;
}

```

Fonction : teste la compatibilité matrimoniale.

Paramètres : deux pointeurs de personnes.

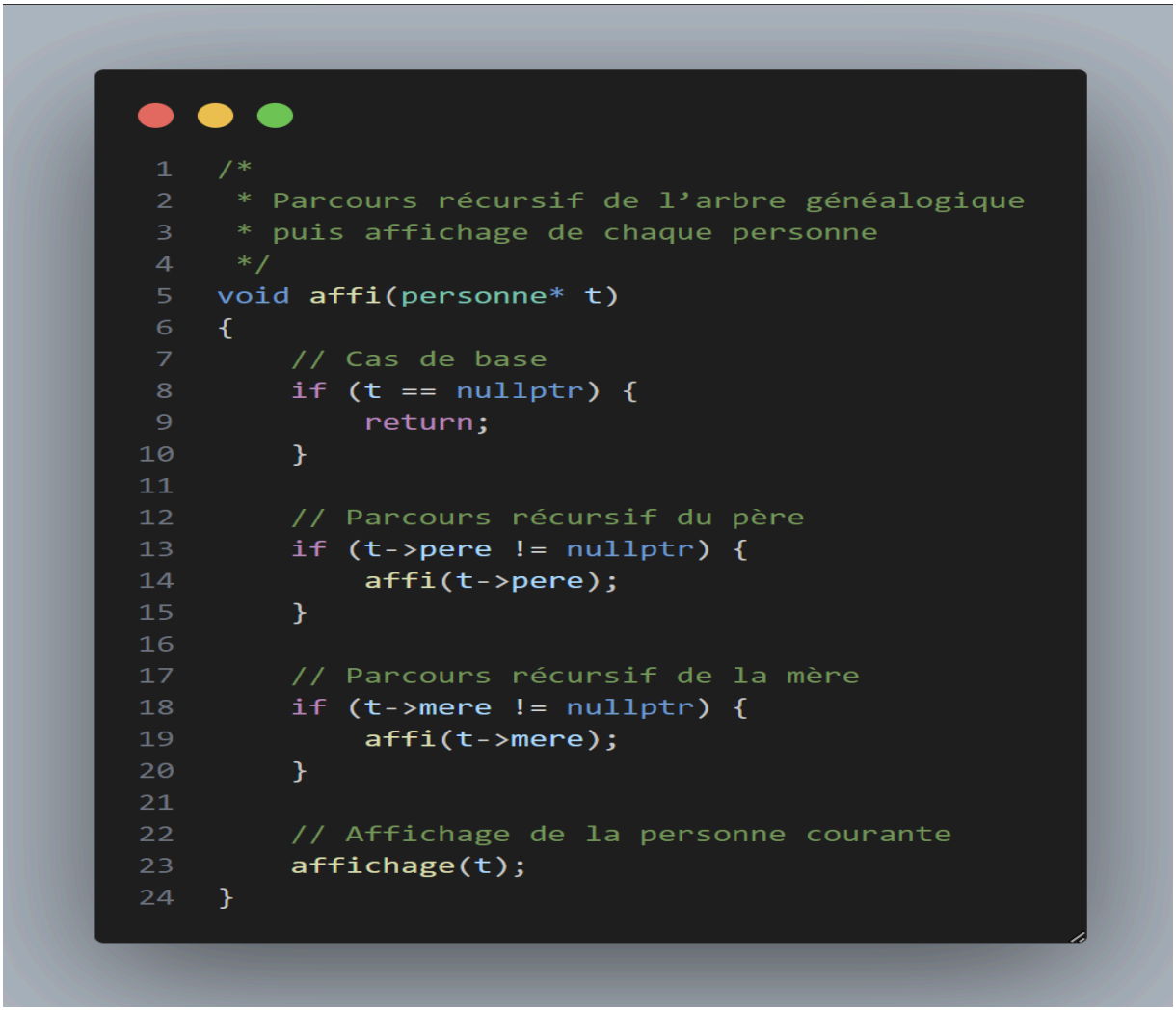
Renvoie : un booléen

fruit :

```
entrez un prenom :
a
entrez un nom :
bl
entrez une date :
1980
entrez un genre 0 si fille, 1 si garcon :
1

entrez un prenom :
b
entrez un nom :
bl
entrez une date :
1980
entrez un genre 0 si fille, 1 si garcon :
0

entrez un prenom :
c
entrez un nom :
vf
entrez une date :
2004
entrez un genre 0 si fille, 1 si garcon :
1

[x] mariage possible
[x] assigniation d'une mere et d'une mere
Prenom : a
Nom : bl
date de naissance :1980
Masculin
recherche amour
pere : b bl
mere : c vf
```

Q10. Affichage de l'arbre généalogique

code :



```
/*
 * Parcours récursif de l'arbre généalogique
 * puis affichage de chaque personne
 */
void affi(personne* t)
{
    // Cas de base
    if (t == nullptr) {
        return;
    }

    // Parcours récursif du père
    if (t->pere != nullptr) {
        affi(t->pere);
    }

    // Parcours récursif de la mère
    if (t->mere != nullptr) {
        affi(t->mere);
    }

    // Affichage de la personne courante
    affichage(t);
}
```

Fonction : affiche une personne et ses parents.

Paramètres : un pointeur vers une personne.

Renvoie : rien.

fruit :

```
===== PARCOURS COMPLET DE L'ARBRE (Depuis Emma) =====
Monsieur Paul Durand epoux de Mme. Martin
De pere inconnu
De mere inconnu
Madame Marie Martin epouse de M.Durand
De pere inconnu
De mere inconnu
Monsieur Jean Durand epoux de Mme. Leroy
De pere Paul Durand
De mere Marie Martin epouse de M.Durand
Madame Sophie Leroy epouse de M.Durand
De pere inconnu
De mere inconnu
Madame Emma Durand pas marrie
De pere Jean Durand
De mere Sophie Leroy epouse de M.Durand
=====================================================
```

Conclusion :

Ce TP nous a permis de manipuler des structures et des pointeurs en C++ à travers la modélisation d'un arbre généalogique. Nous avons appris à créer un type `Personne`, à gérer les relations familiales (mariage, fratrie, ascendance) et à vérifier différentes contraintes logiques. Les fonctions développées nous ont aussi permis d'afficher les informations, de parcourir l'arbre et d'en calculer certaines caractéristiques comme la taille et le nombre de générations. Ce travail nous a aidé à mieux comprendre l'organisation des données, l'utilisation des pointeurs et la programmation récursive.