with Ada.Text\_IO;                   use Ada.Text\_IO;

with Ada.Integer\_Text\_IO;           use Ada.Integer\_Text\_IO;

with Ada.Unchecked\_Deallocation;

package body Registre is

    procedure Free is

        new Ada.Unchecked\_Deallocation (T\_Registre, T\_Access);

            --FONCTIONS/PROCEDURES SECONDAIRES--

    procedure CreateDate(*D*: in out T\_Date; *Jour*: in Integer; *Mois*: in T\_Mois; *Annee*:in Integer) is

        begin

            D.Jour:=Jour;

            D.Mois:=Mois;

            D.Annee:=Annee;

        end CreateDate;

    procedure Afficher\_Date (*Date* : in T\_Date) is

        procedure Afficher\_Deux\_Positions (*Nombre* : in Integer) is

            begin

                Put (Nombre / 10, 1);

                Put (Nombre mod 10, 1);

            end Afficher\_Deux\_Positions; --Pour afficher un jour ou un mois dans le format d'une date

        begin

            Afficher\_Deux\_Positions (Date.Jour);

            Put ('/');

            Afficher\_Deux\_Positions (T\_Mois'pos (Date.Mois) + 1);

            Put ('/');

            Afficher\_Deux\_Positions (Date.Annee / 100);

            Afficher\_Deux\_Positions (abs(Date.Annee) mod 100);

            if Date.Annee<0 then

                Put(" BC");

            end if;

        end Afficher\_Date;

                        --TESTS--

    procedure Init\_RG(*RG*: in out T\_Access) is

        begin

            RG:= new T\_Registre;

        end Init\_RG;

    procedure Start\_RG(*Cle*: in Integer; *Reg*: in out T\_Access) is

        begin

            Reg:= new T\_Registre;

            Reg.all.Cle:=Cle;

        end Start\_RG;

    function Est\_Vide\_RG(*Reg*: in T\_Access) return Boolean is

        begin

            return Reg=Null;

        end Est\_Vide\_RG;

    function Existe\_RG(*Cle*: in Integer;*Reg*: in T\_Access) return Boolean is

        begin

            if Reg=Null then

                return False;

            else

                return (not Est\_Vide\_RG(Rech\_Reg(Cle,Reg)));

            end if;

        end Existe\_RG;

                --FONCTIONS/PROCEDURES ELEMENTAIRES--

    function Name(*Cle*: in Integer; *Reg*:in T\_Access) return Unbounded\_String is

        RegKey: constant T\_Access:=Rech\_Reg(Cle,Reg);

        begin

            if RegKey/=Null then

                return RegKey.all.Nom\_Complet;

            else

                return To\_Unbounded\_String("");

            end if;

        end Name;

    function BirthD(*Cle*: in Integer; *Reg*: in T\_Access) return T\_Date is

        RegKey: constant T\_Access:=Rech\_Reg(Cle,Reg);

        D: T\_Date;

        begin

            if RegKey/=Null then

                return RegKey.all.Date\_Naissance;

            else

                CreateDate(D,01,JANVIER,0000);

                return D;

            end if;

        end BirthD;

    function BirthY(*Cle*: in Integer; *Reg*: in T\_Access) return Integer is

        RegKey: constant T\_Access:=Rech\_Reg(Cle,Reg);

        begin

            if Regkey/=Null then

                return RegKey.all.Date\_Naissance.Annee;

            else

                return 0;

            end if;

        end BirthY;

    function BirthP(*Cle*: in Integer; *Reg*: in T\_Access) return Unbounded\_String is

        RegKey: constant T\_Access:=Rech\_Reg(Cle,Reg);

        begin

            if RegKey/=Null then

                return RegKey.all.Lieu\_Naissance;

            else

                return To\_Unbounded\_String("");

            end if;

        end BirthP;

    procedure RG\_Multiplier\_10(*RG*: in out T\_Access) is

        begin

            if Est\_Vide\_RG(RG) then

                Null;

            else

                RG.all.CLe:=RG.all.Cle\*10;

                RG\_Multiplier\_10(RG.all.Suivant); --Multiplier toutes les clés par 10

            end if;

        end RG\_Multiplier\_10;

                --FONCTIONS/PROCEDURES DE RECHERCHE--

    function Rech\_Reg(*Cle*: in Integer;*Reg*: in T\_Access) return T\_Access is

        Parcours:T\_Access;

        begin

            Parcours:=Reg;

            while Parcours/=Null and then Parcours.all.Cle/=Cle loop --Tant qu'on n'est toujours pas arrivé ni à la clé recherchée ni à la fin du registre

                Parcours:=Parcours.all.Suivant; --Incrémentation du pointeur dans le registre complet

            end loop;

            return Parcours; --Récupérer le registre de la clé

        end Rech\_Reg;

                    --AJOUT/SUPPRESSION--

    procedure AddKey(*Cle*: in Integer; *Reg*: in out T\_Access) is

        Parcours1,Parcours2:T\_Access;

        begin

            if Reg=Null then

                Reg:=new T\_Registre;

                Reg.all.Cle:=Cle; --Nouveau registre initialisé par la Clé Cle

            elsif Rech\_Reg(Cle,Reg)/=Null then

                Null;

            else

                Parcours1:=Reg;

                while Parcours1.all.Suivant/=Null and then Parcours1.all.Suivant.all.Cle<Cle loop --Pour insérer la clé dans le bon endroit afin que les clés soient triées dans l'ordre croissant et assurer une recherche efficace des clés.

                    Parcours1:=Parcours1.all.Suivant;

                end loop;

                Parcours2:=Parcours1.all.Suivant; --Garder le reste du registre

                Parcours1.all.Suivant:=new T\_Registre;

                Parcours1.all.Suivant.all.Cle:=Cle; --Insérer la clé à sa place

                Parcours1.all.Suivant.all.Suivant:=Parcours2; --Rattacher les deux parties du registre

            end if;

        end AddKey;

    procedure Delete\_RG(*Cle*:in Integer; *Reg*: in out T\_Access) is

        Parcours,A\_Supprimer:T\_Access;

        begin

            if Reg=Null then

                Null;

            elsif Rech\_Reg(Cle,Reg)=Null then

                Null;

            else

                if Reg.all.Cle=Cle then

                    Reg:=Reg.all.Suivant;

                else

                    Parcours:=Reg;

                    while Parcours.all.Suivant/= Null and then Parcours.all.Suivant.all.Cle/=Cle loop

                        Parcours:=Parcours.all.Suivant;

                    end loop;

                    A\_Supprimer:=Parcours.all.Suivant;

                    Parcours.all.Suivant:=Parcours.all.Suivant.all.Suivant;

                    Free(A\_Supprimer);

                end if;

            end if;

        end Delete\_RG;

    procedure Detruire\_RG(*RG*: in out T\_Access) is

        begin

            if RG /= Null then

                Detruire\_RG(RG.all.Suivant);

                Free (RG); --Detruction du registre

            else

                Null;

            end if;

        end Detruire\_RG;

                --MODIFICATIONS--

    procedure ModifyKey(*Cle*: in Integer; *NewCle*: in Integer; *Reg*: in out T\_Access) is

        KeyReg:T\_Access;

        begin

            if Reg=Null then

                Reg:=new T\_Registre;

                Reg.all.Cle:=NewCle;

            else

                KeyReg:=Rech\_Reg(Cle,Reg);

                if KeyReg/=Null then

                    KeyReg.all.Cle:=NewCle;

                else

                    Null;

                end if;

            end if;

        end ModifyKey;

    procedure AddName(*Cle*: in Integer; *Nom*: in Unbounded\_String; *Reg*:in out T\_Access) is

        KeyReg:T\_Access;

        begin

            if Reg=Null then

                Reg:= new T\_Registre;

                Reg.all.Cle:=Cle;

                Reg.all.Nom\_Complet:=Nom;

            else

                KeyReg:=Rech\_Reg(Cle,Reg);

                KeyReg.all.Nom\_Complet:=Nom;

            end if;

        end AddName;

    procedure AddBirthD(*Cle*: in Integer; *Jour*: in Integer; *Mois*: in T\_Mois; *Annee*:in Integer; *Reg*: in out T\_Access) is

        KeyReg:T\_Access;

        begin

            if Reg=Null then

                Reg:= new T\_Registre;

                Reg.all.Date\_Naissance.Jour:=Jour;

                Reg.all.Date\_Naissance.Mois:=Mois;

                Reg.all.Date\_Naissance.Annee:=Annee;

                Reg.all.Age:=2019 - Annee;

            else

                KeyReg:=Rech\_Reg(Cle,Reg);

                KeyReg.all.Date\_Naissance.Jour:=Jour;

                KeyReg.all.Date\_Naissance.Mois:=Mois;

                KeyReg.all.Date\_Naissance.Annee:=Annee;

                KeyReg.all.Age:=2019 - Annee;

            end if;

        end AddBirthD;

    procedure AddBirthP(*Cle*: in Integer; *Lieu*: in Unbounded\_String; *Reg*: in out T\_Access) is

        KeyReg:T\_Access;

        begin

            if Reg=Null then

                Reg:= new T\_Registre;

                Reg.all.Lieu\_Naissance:=Lieu;

            else

                KeyReg:=Rech\_Reg(Cle,Reg);

                KeyReg.all.Lieu\_Naissance:=Lieu;

            end if;

        end AddBirthP;

    procedure Ajouter\_Conjoint(*Cle,Conjoint*: in Integer; *RG*: in out T\_Access) is

        KeyReg:T\_Access;

        Ens:arbre\_genealogique.Arbre\_Binaire\_Character.Piles\_Cle.T\_Pile;

        begin

            if RG=Null then

                Put\_Line("Vide!");

            else

                KeyReg:=Rech\_Reg(Cle,RG);

                arbre\_genealogique.Arbre\_Binaire\_Character.Piles\_Cle.Affecter\_Pile(Ens,KeyReg.all.Conjoints);

                arbre\_genealogique.Arbre\_Binaire\_Character.Piles\_Cle.Empiler(Ens,Conjoint);

            end if;

        end Ajouter\_Conjoint;

end Registre;