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

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

with Ada.Strings.Unbounded;         use Ada.Strings.Unbounded;

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

with Arbre\_Genealogique;            use Arbre\_Genealogique;

with Registre;                      use Registre;

procedure main is

    package Piles\_AG renames Arbre\_Genealogique.Arbre\_Binaire\_Character.Piles\_Cle;

    package ABR renames Arbre\_Genealogique.Arbre\_Binaire\_Character;

    AG,NoeudAncetre:ABR.T\_Branch;

    RG:T\_Access;

    Quitter:Boolean:=False;

    MenuPrecedent: Boolean:=False;

    Commencer:Character:='N';

    Addinfo:Character:='N';

    Racine:Integer;

    NbrFils\_Avant,NbrFils\_Apres,g,NewMonth\_Integer,NewDay,NewYear,New\_Key,Newer\_Key,New\_Key\_Desc,Option,Option1,Option2,Option3:Integer;

    New\_Donnee:Character;

    New\_Name,NewBirthP:Unbounded\_String;

    NewMonth:Registre.T\_Mois;

    procedure Start(*Cle*:in Integer;*AG*:in out ABR.T\_Branch;*RG*: in out T\_Access) is

        begin

            Start\_RG(Cle,RG);

            Init\_AG(Cle,AG);

        end Start;

    function Sont\_Vides(*AG*: in ABR.T\_Branch; *RG*: in T\_Access) return Boolean is

        begin

            if Est\_Vide\_RG(RG) and Est\_Nul\_AG(AG) then

                return True;

            elsif (not Est\_Vide\_RG(RG)) and Est\_Nul\_AG(AG) then

                Put\_Line("Seul l'arbre est vide!");

                return False;

            elsif Est\_Vide\_RG(RG) and not Est\_Nul\_AG(AG) then

                Put\_Line("Seul le registre est vide!");

                return False;

            else

                Put\_Line("Ni l'arbre ni le registre est vide.");

                return False;

            end if;

        end Sont\_Vides;

    function Existe(*Cle*: in Integer;*AG*: in out ABR.T\_Branch; *RG*: in T\_Access) return Boolean is

        begin

            if Existe\_RG(Cle,RG) and Est\_Present(Cle,AG) then

                return True;

            elsif (not Existe\_RG(Cle,RG)) and Est\_Present(Cle,AG) then

                Put\_Line("La clé existe dans l'arbre mais pas dans le registre!");

                return False;

            elsif Existe\_RG(Cle,RG) and not Est\_Present(Cle,AG) then

                Put\_Line("La clé existe dans le registre mais pas dans l'arbre!");

                return False;

            else

                Put\_Line("La clé n'existe nul part!");

                return False;

            end if;

        end Existe;

    procedure Edit\_Key(*Cle,NewCle*: in Integer; *AG*: in out ABR.T\_Branch; *RG*: in out T\_Access) is

        CleAncetre,NewParentKey,FGKey,FDKey,TMPKEY,GAUCHEOUDROIT:Integer;

        Noeud: ABR.T\_Branch;

        Ens:Piles\_AG.T\_Pile;

        NewKey:Integer:=NewCle;

        AjoutPossible: Boolean:=False;

        choix: Character:='n';

        begin

            if (not Est\_Nul\_AG(AG)) and (not Est\_Vide\_RG(RG)) then

                if Cle=NewCle then

                    Null;

                elsif NewCle=-181199 or NewCle=-34404 then

                    Put\_Line("Veuillez saisir une autre clé, les clés -34404 et -181199 sont utilisées intérieurement par ce programme pour assurer son fonctionnement..");

                elsif not ABR.Est\_Nul(Rech\_Noeud\_AG(NewCle,AG)) then --Si NewCle existe déjà  dans l'arbre

                    Put\_Line("Existe déjà !");

                else

                    if (not ABR.Est\_Nul(Rech\_Noeud\_AG(Cle,AG))) then

                        if not ABR.Est\_Nul(ABR.Rech\_Ancetre(Cle,AG)) then

                            CleAncetre:=ABR.Nodekey(ABR.Rech\_Ancetre(Cle,AG));

                            if ABR.Nodekey(ABR.Fils\_Gauche(Rech\_Noeud\_AG(CleAncetre,AG)))=Cle then

                                GAUCHEOUDROIT:=CleAncetre-1;

                            else

                                GAUCHEOUDROIT:=CleAncetre+1;

                            end if;

                            Ens:=New\_Key\_Interval(GAUCHEOUDROIT,CleAncetre,AG);

                            if not Est\_Nul\_AG(ABR.Fils\_Droit(Rech\_Noeud\_AG(Cle,AG))) then

                                FDKey:=ABR.Nodekey(ABR.Fils\_Droit(Rech\_Noeud\_AG(Cle,AG)))-1;

                            end if;

                            if not Est\_Nul\_AG(ABR.Fils\_Gauche(Rech\_Noeud\_AG(Cle,AG))) then

                                FGKey:=ABR.Nodekey(ABR.Fils\_Gauche(Rech\_Noeud\_AG(Cle,AG)))+1;

                            end if;

                            if Piles\_AG.Sommet(Ens)=-181199 and Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Ens))=0 and (not Piles\_AG.Est\_Vide(Piles\_AG.Next\_Pile(Piles\_AG.Next\_Pile(Ens)))) and (not Est\_Nul\_AG(ABR.Fils\_Droit(Rech\_Noeud\_AG(Cle,AG)))) then -- intervalle du type [min,+inf[

                                Piles\_AG.Depiler(Ens);Piles\_AG.Depiler(Ens);

                                if (not Est\_Nul\_AG(ABR.Fils\_Gauche(Rech\_Noeud\_AG(Cle,AG)))) and FGKey>Piles\_AG.Sommet(Ens) then

                                    Piles\_AG.Depiler(Ens);Piles\_AG.Empiler(Ens,FGKey);Piles\_AG.Empiler(Ens,FDKey);

                                else

                                    Piles\_AG.Empiler(Ens,FDKey);

                                end if;

                            elsif ((not Est\_Nul\_AG(ABR.Fils\_Gauche(Rech\_Noeud\_AG(Cle,AG)))) and (Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Ens))=0 and (not Piles\_AG.Est\_Vide(Piles\_AG.Next\_Pile(Piles\_AG.Next\_Pile(Ens)))))) and then Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Piles\_AG.Next\_Pile(Ens)))=-181199  then --intervalle du type ]-inf,max]

                                if (not Est\_Nul\_AG(ABR.Fils\_Droit(Rech\_Noeud\_AG(Cle,AG)))) and FDKey<Piles\_AG.Sommet(Ens)then

                                    Piles\_AG.Depiler(Ens);Piles\_AG.Depiler(Ens);Piles\_AG.Depiler(Ens);Piles\_AG.Empiler(Ens,FGKey);Piles\_AG.Empiler(Ens,FDKey);

                                else

                                    TMPKEY:=Piles\_AG.Sommet(Ens);

                                    Piles\_AG.Depiler(Ens);Piles\_AG.Depiler(Ens);Piles\_AG.Depiler(Ens);Piles\_AG.Empiler(Ens,FGKey);Piles\_AG.Empiler(Ens,TMPKEY);

                                end if;

                            elsif Piles\_AG.Est\_Vide(Piles\_AG.Next\_Pile(Piles\_AG.Next\_Pile(Ens))) and Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Ens))<Piles\_AG.Sommet(Ens) and ((not Est\_Nul\_AG(ABR.Fils\_Droit(Rech\_Noeud\_AG(Cle,AG)))) or (not Est\_Nul\_AG(ABR.Fils\_Gauche(Rech\_Noeud\_AG(Cle,AG))))) THEN --intervalle du type [min,max] avec min<max

                                if (not Est\_Nul\_AG(ABR.Fils\_Droit(Rech\_Noeud\_AG(Cle,AG)))) and FDKey<Piles\_AG.Sommet(Ens) then

                                    Piles\_AG.Depiler(Ens);Piles\_AG.Empiler(Ens,FDKey);

                                end if;

                                if (not Est\_Nul\_AG(ABR.Fils\_Gauche(Rech\_Noeud\_AG(Cle,AG)))) and FGKey>Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Ens)) then

                                    TMPKEY:=Piles\_AG.Sommet(Ens);Piles\_AG.Depiler(Ens);Piles\_AG.Depiler(Ens);

                                    Piles\_AG.Empiler(Ens,FGKey);Piles\_AG.Empiler(Ens,TMPKEY);

                                end if;

                            end if;

                            if Piles\_AG.Sommet(Ens)=-181199 and Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Ens))=0 and not Piles\_AG.Est\_Vide(Piles\_AG.Next\_Pile(Piles\_AG.Next\_Pile(Ens))) then -- intervalle du type [min,+inf[

                                while NewKey < Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Piles\_AG.Next\_Pile(Ens))) or (NewKey=-181199 or NewKey=-34404 ) loop

                                    Put("Valeur invalide! Doit être supérieur à  " & Integer'Image(Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Piles\_AG.Next\_Pile(Ens)))) & ". Saisissez une autre valeur : ");

                                    Get(NewKey);

                                    New\_Line;

                                end loop;

                                AjoutPossible:=True;

                            elsif (Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Ens))=0 and (not Piles\_AG.Est\_Vide(Piles\_AG.Next\_Pile(Piles\_AG.Next\_Pile(Ens))))) and then Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Piles\_AG.Next\_Pile(Ens)))=-181199 then --intervalle du type ]-inf,max]

                                while Piles\_AG.Sommet(Ens)<NewKey or (NewKey=-181199 or NewKey=-34404 ) loop

                                    Put("Valeur invalide! Doit être inférieur à  " & Integer'Image(Piles\_AG.Sommet(Ens)) & ". Saisissez une autre valeur : ");

                                    Get(NewKey);

                                    New\_Line;

                                end loop;

                                AjoutPossible:=True;

                            elsif Piles\_AG.Est\_Vide(Piles\_AG.Next\_Pile(Piles\_AG.Next\_Pile(Ens))) and Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Ens))<Piles\_AG.Sommet(Ens) THEN --intervalle du type [min,max] avec min<max

                                while NewKey < Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Ens))  or Piles\_AG.Sommet(Ens) < NewKey or (NewKey=-181199 or NewKey=-34404 )loop

                                    Put("Valeur invalide! Doit être entre " & Integer'Image(Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Ens))) & " et" & Integer'Image(Piles\_AG.Sommet(Ens)) & ". Saisissez une autre valeur : ");

                                    Get(NewKey);

                                    New\_Line;

                                end loop;

                                AjoutPossible:=True;

                            elsif Piles\_AG.Est\_Vide(Piles\_AG.Next\_Pile(Piles\_AG.Next\_Pile(Ens))) and  Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Ens))=Piles\_AG.Sommet(Ens) THEN --une seule valeur possible!

                                while NewKey /= Piles\_AG.Sommet(Ens) loop

                                    Put("Valeur invalide! Doit être égale à " & Integer'Image(Piles\_AG.Sommet(Ens)) & ". Saisissez cette valeur : ");

                                    Get(NewKey);

                                    New\_Line;

                                end loop;

                                AjoutPossible:=True;

                            else --aucune valeur possible!

                                Put\_Line("Modification impossible! Il va falloir modifier la clé: " & Integer'Image(CleAncetre) & " (clé prédécesseur).");

                                Put("Voulez-vous la modifier? [y/n] :"); Get(choix); New\_Line;

                                if choix='n' or choix='N' then

                                    Null;

                                else

                                    Put("Saisissez la valeur de la nouvelle clé que vous voulez attribuer à  " & Integer'Image(CleAncetre) & " : "); Get(NewParentKey); New\_Line;

                                    while NewParentKey=-181199 or NewParentKey=-34404 loop

                                        Put("Veuillez saisir une autre clé, les clés -34404 et -181199 sont utilisées intérieurement par ce programme pour assurer son fonctionnement : ");Get(NewParentKey);

                                        New\_Line;

                                    end loop;

                                    Edit\_Key(CleAncetre, NewParentKey, AG,RG);

                                    Put\_Line("Vous avez modifié la clé du prédécesseur.");

                                    Put\_Line("Vous allez maintenant ressayer de modifier la première clé (" & Integer'Image(NewCle) & " ).");

                                    Edit\_Key(Cle,NewCle,AG,RG);

                                    AjoutPossible:=False;

                                end if;

                            end if;

                            if AjoutPossible then

                                Affecter\_Rech\_Noeud\_AG(Cle,AG,Noeud);

                                Modifier\_Cle\_Racine\_AG(NewKey,Noeud);

                                ModifyKey(Cle,NewKey,RG);

                            else

                                Null;

                            end if;

                        else --Si la clé existe mais n'a pas d'ancêtre (i.e., c'est la racine)

                            Modifier\_Cle\_Racine\_AG(NewKey,AG);

                            ModifyKey(Cle,NewKey,RG);

                        end if;

                    else

                        Put\_Line("Inexistante!");

                    end if;

                end if;

            else

                Start(NewCle,AG,RG);

            end if;

        end Edit\_Key;

    procedure Add\_Ancestor(*Cle\_Nouveau\_Noeud*: in out Integer; *Donnee\_Nouveau\_Noeud*: in Character; *Cle\_Noeud\_Parent*:in integer; *AG*: in out ABR.T\_Branch; *RG*: in out T\_Access) is

        Ens:Piles\_AG.T\_Pile;

        NewKey:Integer:=Cle\_Nouveau\_Noeud;

        AjoutPossible: Boolean:=False;

        choix: Integer;

        NewParentKey:Integer;

        leftright:Character;

        Grand\_Ancestor:Integer;

        begin

            if (Est\_Nul\_AG(AG)) and (Est\_Vide\_RG(RG)) then

                Start(Cle\_Noeud\_Parent, AG, RG);

                Ajouter\_Ancetre(Cle\_Nouveau\_Noeud,Donnee\_Nouveau\_Noeud,Cle\_Noeud\_Parent,AG);

                AddKey(Cle\_Nouveau\_Noeud, RG);

            elsif Cle\_Nouveau\_Noeud=-181199 or Cle\_Nouveau\_Noeud=-34404 then

                Put\_Line("Veuillez saisir une autre clé, les clés -34404 et -181199 sont utilisées intérieurement par ce programme pour assurer son fonctionnement..");

            elsif ABR.Est\_Nul(ABR.Rech\_Noeud(Cle\_Noeud\_Parent,AG)) then --Si le prédécesseur n'existe pas

                Put\_Line("Il faut d'abord créer le prédécesseur!");

            elsif not Est\_Nul\_AG(Rech\_Noeud\_AG(Cle\_Nouveau\_Noeud,AG)) then --Si l'arbre n'est pas vide et que le prédécesseur existe mais que la nouvelle clé existe déjà

                Put\_Line("Existe déjà!");

            elsif (not Est\_Nul\_AG(ABR.Fils\_Droit(Rech\_Noeud\_AG(Cle\_Noeud\_Parent,AG)))) and (not Est\_Nul\_AG(ABR.Fils\_Gauche(Rech\_Noeud\_AG(Cle\_Noeud\_Parent,AG)))) then

                Put\_Line("Plus de place!");

            elsif Cle\_Nouveau\_Noeud>Cle\_Noeud\_Parent and not Est\_Nul\_AG(ABR.Fils\_Droit(Rech\_Noeud\_AG(Cle\_Noeud\_Parent,AG))) then

                if Est\_Nul\_AG(ABR.Fils\_Gauche(Rech\_Noeud\_AG(Cle\_Noeud\_Parent,AG))) then

                    Put\_Line("Emplacement rempli! Ressayez avec une clé inférieure à celle du descendant.");

                else

                    Put\_Line("Plus de place! Tentez plutôt une modification...");

                end if;

            elsif Cle\_Nouveau\_Noeud<Cle\_Noeud\_Parent and not Est\_Nul\_AG(ABR.Fils\_Gauche(Rech\_Noeud\_AG(Cle\_Noeud\_Parent,AG))) then

                if Est\_Nul\_AG(ABR.Fils\_Droit(Rech\_Noeud\_AG(Cle\_Noeud\_Parent,AG))) then

                    Put\_Line("Emplacement rempli! Ressayez avec une clé supérieure à celle du descendant.");

                else

                    Put\_Line("Plus de place! Tentez plutôt une modification...");

                end if;

            else

                 Ens:=New\_Key\_Interval(Cle\_Nouveau\_Noeud,Cle\_Noeud\_Parent,AG);

                 if Piles\_AG.Sommet(Ens)=-181199 and Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Ens))=0 and not Piles\_AG.Est\_Vide(Piles\_AG.Next\_Pile(Piles\_AG.Next\_Pile(Ens))) then

                    while NewKey < Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Piles\_AG.Next\_Pile(Ens))) or (NewKey=-181199 or NewKey=-34404 ) loop

                        Put("Valeur invalide! Doit être supérieur à " & Integer'Image(Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Piles\_AG.Next\_Pile(Ens)))) & ". Saisissez une autre valeur : ");

                        Get(NewKey);

                        New\_Line;

                    end loop;

                    AjoutPossible:=True;

                elsif (Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Ens))=0 and (not Piles\_AG.Est\_Vide(Piles\_AG.Next\_Pile(Piles\_AG.Next\_Pile(Ens))))) and then Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Piles\_AG.Next\_Pile(Ens)))=-181199 then

                    while Piles\_AG.Sommet(Ens)<NewKey or (NewKey=-181199 or NewKey=-34404 ) loop

                        Put("Valeur invalide! Doit être inférieur à " & Integer'Image(Piles\_AG.Sommet(Ens)) & ". Saisissez une autre valeur : ");

                        Get(NewKey);

                        New\_Line;

                    end loop;

                    AjoutPossible:=True;

                elsif Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Ens))<Piles\_AG.Sommet(Ens) THEN

                    while NewKey < Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Ens))  or Piles\_AG.Sommet(Ens) < NewKey or (NewKey=-181199 or NewKey=-34404 ) loop

                        Put("Valeur invalide! Doit être entre " & Integer'Image(Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Ens))) & " et" & Integer'Image(Piles\_AG.Sommet(Ens)) & ". Saisissez une autre valeur : ");

                        Get(NewKey);

                        New\_Line;

                    end loop;

                    AjoutPossible:=True;

                elsif Piles\_AG.Sommet(Piles\_AG.Next\_Pile(Ens))=Piles\_AG.Sommet(Ens) THEN

                    while NewKey /= Piles\_AG.Sommet(Ens) loop

                        Put("Valeur invalide! Doit être égale à " & Integer'Image(Piles\_AG.Sommet(Ens)) & ". Saisissez cette valeur : ");

                        Get(NewKey);

                        New\_Line;

                    end loop;

                    AjoutPossible:=True;

                else

                    Put\_Line("Insertion impossible! Il va falloir modifier la clé: " & Integer'Image(Cle\_Noeud\_Parent) & " (clé prédécesseur) ou multiplier toutes les clés de l'arbre par 10.");

                    Put\_Line("1. Modifier la clé" & Integer'Image(Cle\_Noeud\_Parent));

                    Put\_Line("2. Multiplier toutes les clés de l'arbre par 10");

                    Put("Choisissez une option: "); Get(choix); New\_Line;

                    case choix is

                        when 1=>

                            Put("Saisissez la valeur de la nouvelle clé que vous voulez attribuer à " & Integer'Image(Cle\_Noeud\_Parent) & " : "); Get(NewParentKey); New\_Line;

                            leftright:=ABR.Gauche\_ou\_Droite(Cle\_Noeud\_Parent,AG);

                            if leftright/='R' then

                                Grand\_Ancestor:=ABR.Nodekey(ABR.Rech\_Ancetre(Cle\_Noeud\_Parent,AG));

                            end if;

                            while NewParentKey=-181199 or NewParentKey=-34404 loop

                                Put("Veuillez saisir une autre clé, les clés -34404 et -181199 sont utilisées intérieurement par ce programme pour assurer son fonctionnement : ");Get(NewParentKey);

                                New\_Line;

                            end loop;

                            Edit\_Key(Cle\_Noeud\_Parent, NewParentKey, AG, RG);

                            Put\_Line("Vous allez maintenant ressayer d'ajouter la première clé (" & Integer'Image(NewKey) & " ).");

                            if leftright='G' then

                                NewParentKey:=ABR.Nodekey(ABR.Fils\_Gauche(ABR.Rech\_Noeud(Grand\_Ancestor,AG)));

                            elsif leftright='D' then

                                NewParentKey:=ABR.Nodekey(ABR.Fils\_Droit(ABR.Rech\_Noeud(Grand\_Ancestor,AG)));

                            else

                                NewParentKey:=ABR.Nodekey(AG);

                            end if;

                            Add\_Ancestor(NewKey,Donnee\_Nouveau\_Noeud,NewParentKey,AG,RG);

                        when 2=>

                            ABR.Multiplier\_10(AG);

                            RG\_Multiplier\_10(RG);

                            if Cle\_Nouveau\_Noeud>Cle\_Noeud\_Parent then

                                Put\_Line("La clé" & Integer'Image(NewKey) & " que vous vouliez ajouter à la clé" & Integer'Image(Cle\_Noeud\_Parent\*10) & " est maintenant devenue" & Integer'Image(Cle\_Noeud\_Parent\*10 +5) & ".");

                                Put\_Line("Essai d'ajouter" & Integer'Image(Cle\_Noeud\_Parent\*10 +5) & " à la clé" & Integer'Image(Cle\_Noeud\_Parent\*10) & " :");

                                Cle\_Nouveau\_Noeud:=Cle\_Noeud\_Parent\*10 +5;

                                Add\_Ancestor(Cle\_Nouveau\_Noeud,Donnee\_Nouveau\_Noeud,Cle\_Noeud\_Parent\*10,AG,RG);

                            else

                                Put\_Line("La clé" & Integer'Image(NewKey) & " que vous vouliez ajouter à la clé" & Integer'Image(Cle\_Noeud\_Parent\*10) & " est maintenant devenue" & Integer'Image(Cle\_Noeud\_Parent\*10 -5) & ".");

                                Put\_Line("Essai d'ajouter" & Integer'Image(Cle\_Noeud\_Parent\*10 -5) & " à la clé" & Integer'Image(Cle\_Noeud\_Parent\*10) & " :");

                                Cle\_Nouveau\_Noeud:=Cle\_Noeud\_Parent\*10 -5;

                                Add\_Ancestor(Cle\_Nouveau\_Noeud,Donnee\_Nouveau\_Noeud,Cle\_Noeud\_Parent\*10,AG,RG);

                            end if;

                        when others=> Put\_Line("Option saisie invalide!");

                    end case;

                end if;

                if AjoutPossible then

                    Ajouter\_Ancetre(NewKey,Donnee\_Nouveau\_Noeud,Cle\_Noeud\_Parent,AG);

                    AddKey(NewKey,RG);

                    Cle\_Nouveau\_Noeud:=NewKey;

                else

                    Null;

                end if;

            end if;

        end Add\_Ancestor;

    procedure Edit\_Sexe(*Cle*: in Integer; *Value*: in Character; *AG*: in out ABR.T\_Branch) is

        begin

            Modifier\_Sexe\_AG(Cle,Value,AG);

        end Edit\_Sexe;

    procedure Add\_Name(*Cle*: in Integer; *Nom*: in Unbounded\_String; *RG*:in out T\_Access) is

        begin

            AddName(Cle,Nom,RG);

        end Add\_Name;

    procedure Add\_BirthD(*Cle*: in Integer; *Jour*: in Integer; *Mois*: in T\_Mois; *Annee*:in Integer; *RG*: in out T\_Access) is

        begin

            AddBirthD(Cle,Jour,Mois,Annee,RG);

        end Add\_BirthD;

    procedure Add\_BirthP(*Cle*: in Integer; *Lieu*: in Unbounded\_String; *RG*: in out T\_Access) is

        begin

            AddBirthP(Cle,Lieu,RG);

        end Add\_BirthP;

    function Full\_Name(*Cle*: in Integer; *RG*: in T\_Access) return Unbounded\_String is

        begin

            return Name(Cle,RG);

        end Full\_Name;

    function Birth\_Date(*Cle*: in Integer; *RG*: in T\_Access) return T\_Date is

        begin

            return BirthD(Cle,RG);

        end Birth\_Date;

    function Birth\_Year(*Cle*: in Integer; *RG*: in T\_Access) return Integer is

        begin

            return BirthY(Cle,RG);

        end Birth\_Year;

    function Birth\_Place(*Cle*: in Integer; *RG*: in T\_Access) return Unbounded\_String is

        begin

            return BirthP(Cle,RG);

        end Birth\_Place;

    procedure Delete(*Cle*: in Integer; *AG*: in out ABR.T\_Branch; *RG*: in out T\_Access) is

        Ens: Piles\_AG.T\_Pile;

        begin

            if Existe(Cle,AG,RG) then

                Ens:=ABR.Ensemble\_Fils\_Noeud(Cle,AG); --Les ancêtres de la clé Cle

                while not Piles\_AG.Est\_Vide(Ens) loop --Tant que la pile des des ancêtres de Cle n'est pas vide

                    Delete\_RG(Piles\_AG.Sommet(Ens),RG); --Supprimer un des ancêtres de la clé du registre

                    Piles\_AG.Depiler(Ens);              --Dépiler la pile

                end loop; --La boucle finit quand tous les ancêtres de Cle ont été supprimés du registre

                --Le traitement précédent qui supprime tous les ancêtres d'une clé du registre ne figure pas dans le module Registre parce qu'il a besoin de l'arbre généalogique pour reconnaître les ancêtres de cette clé

                Delete\_RG(Cle,RG);                      --Supprimer la clé Cle du registre

                Supprimer\_Famille(Cle,AG);              --Supprimer la clé Cle et tous ses ancêtres de l'arbre

            else

                Null;

            end if;

        end Delete;

    function Ancestor\_Nbr(*Cle*: in Integer; *AG*: in ABR.T\_Branch) return Integer is

        begin

            return Nombre\_Ancetres(Cle,AG);

        end Ancestor\_Nbr;

    procedure Same\_Gen\_Keys(*g,Cle*:in Integer; *AG*: in ABR.T\_Branch) is

        begin

            Ensemble\_Ancetres\_Meme\_Generation(g,Cle,AG);

        end Same\_Gen\_Keys;

    procedure Same\_Gen\_Orless(*g,Cle*: in Integer; *AG*: in ABR.T\_Branch) is

        begin

            Ensemble\_Ancetres\_Generation\_N(g,Cle,AG);

        end Same\_Gen\_Orless;

    procedure Print\_From(*Cle*: in Integer; *AG*: in ABR.T\_Branch) is

        begin

            Afficher\_A\_Partir(Cle,AG);

        end Print\_From;

    procedure Print(*AG*: in ABR.T\_Branch) is

        begin

            Afficher\_AG(AG);

        end Print;

    function Homonymes(*m1,m2*: in Integer; *AG*: in ABR.T\_Branch; *RG*: in T\_Access) return Boolean is

        Node1:constant ABR.T\_Branch:=Rech\_Noeud\_AG(m1,AG); --Noeud de m1

        Node2:constant ABR.T\_Branch:=Rech\_Noeud\_AG(m2,AG); --Noeud de m2

        B1,B2:Boolean :=False;                             -- Booléens qui vont déterminer si un homonyme a été trouvé

        procedure Intermediaire1(*Parcours1,Parcours2*: in ABR.T\_Branch; *AG*: in ABR.T\_Branch; *RG*: in T\_Access) is

            begin

                if not Est\_Nul\_AG(Parcours2) then

                    if Full\_Name(ABR.Nodekey(Parcours2),RG)/=To\_Unbounded\_String("") and then Full\_Name(ABR.Nodekey(Parcours2),RG)=Full\_Name(ABR.Nodekey(Parcours1),RG) then

                    --Si le nom de la racine de Parcours 2 n'est pas nul et qu'il est égal à celui de la racine de Parcours 1

                        B2:=True; --B2 est vrai est signifiera que l'homonyme a été retrouvé

                    else

                        if not Est\_Nul\_AG(ABR.Fils\_Gauche(Parcours2)) then --Si le fils gauche de Parcours 2 n'est pas nul

                            Intermediaire1(Parcours1,ABR.Fils\_Gauche(Parcours2),AG,RG); --On recherche dans le fils gauche de Parcours 2

                        end if;

                        if not Est\_Nul\_AG(ABR.Fils\_Droit(Parcours2)) then --Si le fils droit de Parcours 2 n'est pas nul

                            Intermediaire1(Parcours1,ABR.Fils\_Droit(Parcours2),AG,RG); --On recherche dans le fils droit de Parcours 2

                        end if;

                    end if;

                end if;

            end Intermediaire1;

        procedure Intermediaire2(*Parcours1*: in ABR.T\_Branch; *AG*: in ABR.T\_Branch; *RG*: in T\_Access) is

            begin

                if not Est\_Nul\_AG(AG) then

                    Intermediaire1(Parcours1,Node2,AG,RG); --Voir si le nom de la racine de Parcours 1 figure quelque part dans tout le sous-arbre de m2

                    if B2 then --Si B2 est devenu True alors l'homonyme a été trouvé

                        B1:=True; --B1 signifiera maintenant que l'homonyme a été trouvé

                    else

                        if (not Est\_Nul\_AG(ABR.Fils\_Gauche(Parcours1))) then --Si le fils gauche de Parcours 1 n'est pas nul

                            Intermediaire2(ABR.Fils\_Gauche(Parcours1),AG,RG); --On teste maintenant si le nom du fils gauche figure quelquepart dans le sous-arbre de m2

                        end if;

                        if (not Est\_Nul\_AG(ABR.Fils\_Droit(Parcours1))) then --Si le fils droit de Parcours 1 n'est pas nul

                            Intermediaire2(ABR.Fils\_Droit(Parcours1),AG,RG); --On teste si le nom du fils droit figure quelquepart dans le sous-arbre de m2

                        end if;

                    end if;

                end if;

            end Intermediaire2;

        begin

            Intermediaire2(Node1,AG,RG);

            if B1 then return True; --Si B1 est devenu vrai c'est qu'au moins un homonyme existe

            else return False;      --Sinon, il n'y a pas d'homonymes

            end if;

        end Homonymes;

    procedure Devtest(*AG*: in out ABR.T\_Branch; *RG*: in out T\_Access) is

        int: Integer;

        begin

            Start(20,AG,RG);

            int:=10 ;Add\_Ancestor(int,'P',20,AG,RG);

            int:= 9 ;Add\_Ancestor(int,'P',10,AG,RG);

            int:=15 ;Add\_Ancestor(int,'M',10,AG,RG);

            int:=13 ;Add\_Ancestor(int,'P',15,AG,RG);

            int:=12 ;Add\_Ancestor(int,'P',13,AG,RG);

            int:=14 ;Add\_Ancestor(int,'M',13,AG,RG);

            int:=17 ;Add\_Ancestor(int,'M',15,AG,RG);

            int:=16 ;Add\_Ancestor(int,'P',17,AG,RG);

            int:=19 ;Add\_Ancestor(int,'M',17,AG,RG);

            int:=18 ;Add\_Ancestor(int,'P',19,AG,RG);

            int:=30 ;Add\_Ancestor(int,'M',20,AG,RG);

            int:=35 ;Add\_Ancestor(int,'M',30,AG,RG);

            int:=32 ;Add\_Ancestor(int,'P',35,AG,RG);

            int:=31 ;Add\_Ancestor(int,'P',32,AG,RG);

            int:=33 ;Add\_Ancestor(int,'M',32,AG,RG);

            int:=34 ;Add\_Ancestor(int,'M',33,AG,RG);

            int:=37 ;Add\_Ancestor(int,'M',35,AG,RG);

            int:=36 ;Add\_Ancestor(int,'P',37,AG,RG);

            int:=38 ;Add\_Ancestor(int,'M',37,AG,RG);

            Add\_Name(19,To\_Unbounded\_String("Bob"),RG);

            Add\_Name(36,To\_Unbounded\_String("Bob"),RG);

        end Devtest;

    begin

        Put\_Line(" ");

        Put\_Line("          Arbres Généalogiques et Registres d'Etat Civil          ");New\_Line;New\_Line;

        Put("Voulez-vous créer un arbre généalogique et son registre ? [y/n] : ");

        Get(Commencer); New\_Line; --traiter l'erreur où l'utilisateur saisit autre chose!!!

        if Commencer/='n' and Commencer/='N' then

            Skip\_Line;

            Put("Commencez d'abord par saisir la valeur de la racine (nombre entier) : "); --traiter l'erreur où l'utilisateur saisit autre chose!!!

            Get(Racine);New\_Line;

            Start(Racine,AG,RG); New\_Line;

            Put("Saisissez le nom complet de la racine : ");Skip\_Line;New\_Name:=To\_Unbounded\_String(Get\_Line);New\_Line;

            Add\_Name(Racine,New\_Name,RG);

            Put\_Line("Saisissez sa date de naissance : ");

            Put("      -Le jour : "); Get(NewDay,2);New\_Line;Skip\_Line;

            while NewDay<1 or NewDay>31 loop

                Put("       Le jour saisi est invalide! Saisissez une valeur entre 1 et 31 : ");Get(NewDay);New\_Line;Skip\_Line;

            end loop;

            Put("      -Le mois : "); Get(NewMonth\_Integer,2);New\_Line;Skip\_Line;

            while NewMonth\_Integer<1 or NewMonth\_Integer>12 loop

                Put("       Le mois saisi est invalide! Saisissez une valeur entre 1 et 12 : ");Get(NewMonth\_Integer);New\_Line;Skip\_Line;

            end loop;

            Put("      -L'année : "); Get(NewYear);New\_Line;

            case NewMonth\_Integer is

                when 1=>NewMonth:=JANVIER;Add\_BirthD(Racine,NewDay,NewMonth,NewYear,RG);

                when 2=>NewMonth:=FEVRIER;Add\_BirthD(Racine,NewDay,NewMonth,NewYear,RG);

                when 3=>NewMonth:=MARS;Add\_BirthD(Racine,NewDay,NewMonth,NewYear,RG);

                when 4=>NewMonth:=AVRIL;Add\_BirthD(Racine,NewDay,NewMonth,NewYear,RG);

                when 5=>NewMonth:=MAI;Add\_BirthD(Racine,NewDay,NewMonth,NewYear,RG);

                when 6=>NewMonth:=JUIN;Add\_BirthD(Racine,NewDay,NewMonth,NewYear,RG);

                when 7=>NewMonth:=JUILLET;Add\_BirthD(Racine,NewDay,NewMonth,NewYear,RG);

                when 8=>NewMonth:=AOUT;Add\_BirthD(Racine,NewDay,NewMonth,NewYear,RG);

                when 9=>NewMonth:=SEPTEMBRE;Add\_BirthD(Racine,NewDay,NewMonth,NewYear,RG);

                when 10=>NewMonth:=OCTOBRE;Add\_BirthD(Racine,NewDay,NewMonth,NewYear,RG);

                when 11=>NewMonth:=NOVEMBRE;Add\_BirthD(Racine,NewDay,NewMonth,NewYear,RG);

                when 12=>NewMonth:=DECEMBRE;Add\_BirthD(Racine,NewDay,NewMonth,NewYear,RG);

                when others=>

                    Put\_Line("Le mois saisi est invalide!");

            end case;

            Put("Saisissez son lieu de naissance : ");Skip\_Line;NewBirthP:=To\_Unbounded\_String(Get\_Line); New\_Line; New\_Line;

            Add\_BirthP(Racine,NewBirthP,RG);

            Put\_Line("Vous avez bien construit la racine de l'arbre généalogique.");New\_Line;New\_Line;

            loop

                Put\_Line("         Menu Principal           ");New\_Line;

                Put\_Line("0. DEV TEST");

                Put\_Line("1. Vérifications");

                Put\_Line("2. Opérations");

                Put\_Line("3. Affichages");

                Put\_Line("4. Quitter ?");New\_Line;

                Put("Quelle option choisissez vous? : ");

                Get(Option);New\_Line;

                New\_Line;

                case Option is

                    when 0 =>

                        Devtest(AG,RG); Racine:=20;

                        Put\_Line("Arbre de DEVTEST construit.");New\_Line;New\_Line;

                    when 1 =>

                        loop

                            Put\_Line("         Menu des Vérifications          ");New\_Line;

                            Put\_Line("1. Vérifier si l'arbre et le registre sont vides");

                            Put\_Line("2. Vérifier si une clé existe dans l'arbre et son registre");

                            Put\_Line("3. Vérifier si deux individus ont un ou plusieurs ancêtres homonymes");

                            Put\_Line("4. Revenir au Menu Principal");New\_Line;

                            Put("Quelle option choisissez vous? : ");

                            Get(Option1); New\_Line;New\_Line;

                            case Option1 is

                                when 1 =>

                                    if Sont\_Vides(AG,RG) then Put\_Line("L'arbre et son registre sont vides."); else Null; end if;

                                when 2 =>

                                    Put("Saisissez la clé que vous cherchez : "); Get(New\_Key);New\_Line;

                                    if Existe(New\_Key,AG,RG) then Put\_Line("La clé existe dans l'arbre et le registre."); else Null; end if; New\_Line;New\_Line;

                                when 3 =>

                                    Put("Saisissez la clé du premier individu : "); Get(New\_Key);New\_Line;

                                    Put("Saisissez la clé du deuxième individu : ");Get(Newer\_Key);New\_Line;

                                    if not Existe(New\_Key,AG,RG) or not Existe(Newer\_Key,AG,RG) then

                                        Put\_Line("Clé(s) introuvable(s) !"); New\_Line; New\_Line;

                                    else

                                        if Homonymes(New\_Key,Newer\_Key,AG,RG) then

                                            Put\_Line("Ces individus ont bien un ou plusieurs ancêtres homonymes."); New\_Line; New\_Line;

                                        else

                                            Put\_Line("Ces individus n'ont pas d'ancêtres homonymes."); New\_Line; New\_Line;

                                        end if;

                                    end if;

                                when 4 =>

                                    MenuPrecedent:=True;

                                    Put\_Line("Retour au menu précédent..."); New\_Line;New\_Line;

                                when others =>

                                    Put\_Line("Saisissez une option valide."); New\_Line;

                            end case;

                            New\_Line;

                        exit when MenuPrecedent;

                        end loop;

                        MenuPrecedent:=False;

                    when 2 =>

                        loop

                            Put\_Line("         Menu des Opérations           ");New\_Line;

                            Put\_Line("0. Multiplier toutes les clés par 10");

                            Put\_Line("1. Ajouter un ancêtre");

                            Put\_Line("2. Supprimer un ancêtre");

                            Put\_Line("3. Modifier la clé d'un ancêtre");

                            Put\_Line("4. Modifier le sexe d'un ancêtre");

                            Put\_Line("5. Ajouter/Modifier toutes les informations d'un ancêtre");

                            Put\_Line("6. Ajouter/Modifier le nom complet d'un ancêtre");

                            Put\_Line("7. Ajouter/Modifier la date de naissance d'un ancêtre");

                            Put\_Line("8. Ajouter/Modifier le lieu de naissance d'un ancêtre");

                            Put\_Line("9. Revenir au Menu Principal");New\_Line;

                            Put("Quelle option choisissez vous? : ");

                            Get(Option2); New\_Line; New\_Line;

                            case Option2 is

                                when 0=>

                                    ABR.Multiplier\_10(AG);

                                    RG\_Multiplier\_10(RG);

                                    Racine:=Racine\*10;

                                    Put\_Line("Toutes les clés ont été multipliées par 10.");New\_Line;New\_Line;

                                when 1 =>

                                    Put("Saisissez la clé du nouvel ancêtre : "); Get(New\_Key);New\_Line;

                                    Put("Saisissez la clé de son descendant : "); Get(New\_Key\_Desc);New\_Line;

                                    Put("Saisissez le lien de parenté [M/P]: "); Get(New\_Donnee);New\_Line;

                                    NbrFils\_Avant:=Nombre\_Ancetres(New\_Key\_Desc,AG);

                                    Affecter\_Rech\_Noeud\_AG(New\_Key\_Desc,AG,NoeudAncetre);

                                    Add\_Ancestor(New\_Key,New\_Donnee,New\_Key\_Desc,AG,RG);

                                    Put("Vérification que l'ajout a été fait... ");

                                    if ABR.Nodekey(AG)=10\*Racine then

                                        Racine:=ABR.Nodekey(AG);

                                        if New\_Key>New\_Key\_Desc then

                                            New\_Key:=(New\_Key\_Desc\*10)+5;

                                        else

                                            New\_Key:=(New\_Key\_Desc\*10)-5;

                                        end if;

                                    end if;

                                    NbrFils\_Apres:=Nombre\_Ancetres(ABR.Nodekey(NoeudAncetre),AG);

                                    if NbrFils\_Avant=NbrFils\_Apres-1 then

                                        Put("Ajout réussi.");New\_Line;

                                        Put("Voulez-vous attribuer à la clé des informations ? [y/n] : "); Get(AddInfo);New\_Line;

                                        if AddInfo/='n' and AddInfo/='N' then

                                            Put("Saisissez le nom complet du nouvel ancêtre : ");Skip\_Line;New\_Name:=To\_Unbounded\_String(Get\_Line);New\_Line;

                                            Add\_Name(New\_Key,New\_Name,RG);

                                            Put\_Line("Saisissez sa date de naissance : ");

                                            Put("      -Le jour : "); Get(NewDay);New\_Line;Skip\_Line;

                                            while NewDay<1 or NewDay>31 loop

                                                Put("       Le jour saisi est invalide! Saisissez une valeur entre 1 et 31 : ");Get(NewDay);New\_Line;Skip\_Line;

                                            end loop;

                                            Put("      -Le mois : "); Get(NewMonth\_Integer);New\_Line;Skip\_Line;

                                            while NewMonth\_Integer<1 or NewMonth\_Integer>12 loop

                                                Put("       Le mois saisi est invalide! Saisissez une valeur entre 1 et 12 : ");Get(NewMonth\_Integer);New\_Line;Skip\_Line;

                                            end loop;

                                            Put("      -L'année : "); Get(NewYear);New\_Line;

                                            while (Birth\_Year(New\_Key\_Desc,RG) - NewYear)<16 loop

                                                Put\_Line("Un parent doit être plus âgé que son descendant d'au moins 16 ans!");

                                                Put("Saisissez une année positive et inférieure ou égale à" & Integer'Image(Birth\_Year(New\_Key\_Desc,RG)-16) & " : "); Get(NewYear);New\_Line;

                                            end loop;

                                            case NewMonth\_Integer is

                                                when 1=>NewMonth:=JANVIER;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                                when 2=>NewMonth:=FEVRIER;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                                when 3=>NewMonth:=MARS;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                                when 4=>NewMonth:=AVRIL;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                                when 5=>NewMonth:=MAI;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                                when 6=>NewMonth:=JUIN;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                                when 7=>NewMonth:=JUILLET;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                                when 8=>NewMonth:=AOUT;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                                when 9=>NewMonth:=SEPTEMBRE;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                                when 10=>NewMonth:=OCTOBRE;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                                when 11=>NewMonth:=NOVEMBRE;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                                when 12=>NewMonth:=DECEMBRE;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                                when others=>

                                                    Put\_Line("Le mois saisi est invalide!");

                                            end case;

                                            Put("Saisissez son lieu de naissance : ");Skip\_Line;NewBirthP:=To\_Unbounded\_String(Get\_Line); New\_Line; New\_Line;

                                            Add\_BirthP(New\_Key,NewBirthP,RG);

                                            Put\_Line("Vous avez bien ajouté le nouvel ancêtre et ses informations à l'arbre et au registre.");New\_Line;New\_Line;

                                        else

                                            Put\_Line("Retour au menu des opérations...");New\_Line;New\_Line;

                                        end if;

                                    else

                                        Put\_Line("Ajout échoué!");New\_Line;New\_Line;

                                    end if;

                                when 2 =>

                                    Put("Saisissez la clé de l'ancêtre que vous voulez supprimer : "); Get(New\_Key);New\_Line;

                                    Delete(New\_Key,AG,RG);

                                    if not (Est\_Present(New\_Key,AG) and Existe\_RG(New\_Key,RG)) then

                                        Put\_Line("Suppression effectuée."); New\_Line;New\_Line;

                                    else

                                        Put\_Line("Suppression échouée!"); New\_Line;New\_Line;

                                    end if;

                                when 3 =>

                                    Put("Saisissez la clé de l'ancêtre dont vous voulez changer la clé : "); Get(New\_Key);New\_Line;

                                    if Existe(New\_Key,AG,RG) then

                                        Put("Saissez la nouvelle clé : ");Get(Newer\_Key);New\_Line;

                                        Edit\_Key(New\_Key,Newer\_Key,AG,RG);

                                        Put\_Line("Modification effectuée.");

                                        if New\_Key=Racine then

                                            Racine:=ABR.Nodekey(AG);

                                        end if;

                                    end if;

                                    New\_Line;New\_Line;

                                when 4 =>

                                    Put("Saisissez la clé de l'ancêtre dont vous voulez changer le sexe : "); Get(New\_Key);New\_Line;

                                    if Existe(New\_Key,AG,RG) then

                                        Put("Saisissez le nouveau sexe [M/P]: "); Get(New\_Donnee);New\_Line; --control error!

                                        Edit\_Sexe(New\_Key,New\_Donnee,AG);

                                        Put\_Line("Modification effectuée.");

                                    end if;

                                    New\_Line;New\_Line;

                                when 5 =>

                                    Put("Saisissez la clé de l'ancêtre dont vous voulez attribuer des informations : "); Get(New\_Key);New\_Line;

                                    if Existe(New\_Key,AG,RG) then

                                        Put("Saisissez le nom complet de l'ancêtre : ");Skip\_Line;New\_Name:=To\_Unbounded\_String(Get\_Line);New\_Line;

                                        Add\_Name(New\_Key,New\_Name,RG);

                                        Put\_Line("Saisissez sa date de naissance : ");

                                        Put("      -Le jour : "); Get(NewDay);New\_Line;Skip\_Line;

                                        while NewDay<1 or NewDay>31 loop

                                                Put("       Le jour saisi est invalide! Saisissez une valeur entre 1 et 31 : ");Get(NewDay);New\_Line;Skip\_Line;

                                            end loop;

                                            Put("      -Le mois : "); Get(NewMonth\_Integer);New\_Line;Skip\_Line;

                                            while NewMonth\_Integer<1 or NewMonth\_Integer>12 loop

                                                Put("       Le mois saisi est invalide! Saisissez une valeur entre 1 et 12 : ");Get(NewMonth\_Integer);New\_Line;Skip\_Line;

                                            end loop;

                                        Put("      -L'année : "); Get(NewYear);New\_Line;

                                        while New\_Key/=(ABR.Nodekey(AG)) and then (Birth\_Year(ABR.Nodekey(ABR.Rech\_Ancetre(New\_Key,AG)),RG) - NewYear)<16 loop

                                            Put\_Line("Un parent doit être plus âgé que son descendant d'au moins 16 ans!");

                                            Put("Saisissez une année positive et inférieure ou égale à" & Integer'Image(Birth\_Year(ABR.Nodekey(ABR.Rech\_Ancetre(New\_Key,AG)),RG)-16) & " : "); Get(NewYear);New\_Line;

                                        end loop;

                                        case NewMonth\_Integer is

                                            when 1=>NewMonth:=JANVIER;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                            when 2=>NewMonth:=FEVRIER;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                            when 3=>NewMonth:=MARS;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                            when 4=>NewMonth:=AVRIL;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                            when 5=>NewMonth:=MAI;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                            when 6=>NewMonth:=JUIN;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                            when 7=>NewMonth:=JUILLET;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                            when 8=>NewMonth:=AOUT;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                            when 9=>NewMonth:=SEPTEMBRE;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                            when 10=>NewMonth:=OCTOBRE;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                            when 11=>NewMonth:=NOVEMBRE;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                            when 12=>NewMonth:=DECEMBRE;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);

                                            when others=>

                                                Put\_Line("Le mois saisi est invalide!");

                                        end case;

                                        Put("Saisissez son lieu de naissance : ");Skip\_Line;NewBirthP:=To\_Unbounded\_String(Get\_Line); New\_Line; New\_Line;

                                        Add\_BirthP(New\_Key,NewBirthP,RG);

                                        Put\_Line("Vous avez bien ajouté/modifié toutes les informations de l'ancêtre.");

                                    end if;

                                    New\_Line;New\_Line;

                                when 6 =>

                                    Put("Saisissez la clé de l'ancêtre auquel vous voulez attribuer un nom : "); Get(New\_Key);New\_Line;

                                    if Existe(New\_Key,AG,RG) then

                                        Put("Saisissez le nom complet : ");Skip\_Line; New\_Name:=To\_Unbounded\_String(Get\_Line);New\_Line; --control error!

                                        Add\_Name(New\_Key,New\_Name,RG);

                                        Put\_Line("Ajout/Modification effectuée.");

                                    end if;

                                    New\_Line;New\_Line;

                                when 7 =>

                                    Put("Saisissez la clé de l'ancêtre auquel vous voulez attribuer une date de naissance : ");Get(New\_Key);New\_Line;

                                    if Existe(New\_Key,AG,RG) then

                                        Put\_Line("Saisissez sa date de naissance : "); Skip\_Line;

                                        Put("      -Le jour : "); Get(NewDay);New\_Line;Skip\_Line;

                                        while NewDay<1 or NewDay>31 loop

                                                Put("       Le jour saisi est invalide! Saisissez une valeur entre 1 et 31 : ");Get(NewDay);New\_Line;Skip\_Line;

                                            end loop;

                                            Put("      -Le mois : "); Get(NewMonth\_Integer);New\_Line;Skip\_Line;

                                            while NewMonth\_Integer<1 or NewMonth\_Integer>12 loop

                                                Put("       Le mois saisi est invalide! Saisissez une valeur entre 1 et 12 : ");Get(NewMonth\_Integer);New\_Line;Skip\_Line;

                                            end loop;

                                        Put("      -L'année : "); Get(NewYear);New\_Line;

                                        while New\_Key/=ABR.Nodekey(AG) and then (Birth\_Year(ABR.Nodekey(ABR.Rech\_Ancetre(New\_Key,AG)),RG) - NewYear)<16 loop

                                                Put\_Line("Un parent doit être plus âgé que son descendant d'au moins 16 ans!");

                                                Put("Saisissez une année positive et inférieure ou égale à" & Integer'Image(Birth\_Year(ABR.Nodekey(ABR.Rech\_Ancetre(New\_Key,AG)),RG)-16) & " : "); Get(NewYear);New\_Line;

                                        end loop;

                                        case NewMonth\_Integer is

                                            when 1=>NewMonth:=JANVIER;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);Put\_Line("Ajout/Modification effectuée.");

                                            when 2=>NewMonth:=FEVRIER;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);Put\_Line("Ajout/Modification effectuée.");

                                            when 3=>NewMonth:=MARS;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);Put\_Line("Ajout/Modification effectuée.");

                                            when 4=>NewMonth:=AVRIL;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);Put\_Line("Ajout/Modification effectuée.");

                                            when 5=>NewMonth:=MAI;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);Put\_Line("Ajout/Modification effectuée.");

                                            when 6=>NewMonth:=JUIN;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);Put\_Line("Ajout/Modification effectuée.");

                                            when 7=>NewMonth:=JUILLET;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);Put\_Line("Ajout/Modification effectuée.");

                                            when 8=>NewMonth:=AOUT;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);Put\_Line("Ajout/Modification effectuée.");

                                            when 9=>NewMonth:=SEPTEMBRE;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);Put\_Line("Ajout/Modification effectuée.");

                                            when 10=>NewMonth:=OCTOBRE;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);Put\_Line("Ajout/Modification effectuée.");

                                            when 11=>NewMonth:=NOVEMBRE;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);Put\_Line("Ajout/Modification effectuée.");

                                            when 12=>NewMonth:=DECEMBRE;Add\_BirthD(New\_Key,NewDay,NewMonth,NewYear,RG);Put\_Line("Ajout/Modification effectuée.");

                                            when others=>

                                                Put\_Line("Le mois saisi est invalide!");

                                        end case;

                                    end if;

                                    New\_Line;New\_Line;

                                when 8 =>

                                    Put("Saisissez la clé de l'ancêtre auquel vous voulez attribuer un lieu de naissance : ");Get(New\_Key);New\_Line;

                                    if Existe(New\_Key,AG,RG) then

                                        Put("Saisissez son lieu de naissance : ");Skip\_Line; NewBirthP:=To\_Unbounded\_String(Get\_Line);New\_Line;

                                        Add\_BirthP(New\_Key,NewBirthP,RG);

                                        Put\_Line("Ajout/Modification effectuée.");

                                    end if;

                                    New\_Line;New\_Line;

                                when 9 =>

                                    MenuPrecedent:=True;

                                    Put\_Line("Retour au menu précédent..."); New\_Line;New\_Line;

                                when others =>

                                    Put\_Line("Saisissez une option valide."); New\_Line;

                            end case;

                        exit when MenuPrecedent;

                        end loop;

                        MenuPrecedent:=False;

                    when 3 =>

                        loop

                            Put\_Line("         Menu des Affichages           ");New\_Line;

                            Put\_Line(" 1. Afficher toutes les informations d'une clé");

                            Put\_Line(" 2. Afficher le nom complet d'une clé");

                            Put\_Line(" 3. Afficher la date de naissance d'une clé");

                            Put\_Line(" 4. Afficher le lieu de naissance d'une clé");

                            Put\_Line(" 5. Afficher les clés et noms des parents d'une clé");

                            Put\_Line(" 6. Afficher le nombre d'ancêtres d'une clé");

                            Put\_Line(" 7. Afficher les clés d'une certaine génération par rapport à une clé");

                            Put\_Line(" 8. Afficher les clés d'une certaine génération ou moins par rapport à une clé");

                            Put\_Line(" 9. Afficher les individus n'ayant aucun parent connu");

                            Put\_Line("10. Afficher les individus ayant un seul parent connu");

                            Put\_Line("11. Afficher les individus dont les deux parents sont connus");

                            Put\_Line("12. Afficher l'arbre complet");

                            Put\_Line("13. Afficher l'arbre à partir d'une clé");

                            Put\_Line("14. Revenir au Menu Principal");New\_Line;

                            Put("Quelle option choisissez vous? : ");

                            Get(Option3); New\_Line;New\_Line;

                            case Option3 is

                                when 1=>

                                    Put("Saisissez la clé de l'ancêtre dont vous voulez afficher toutes les informations : "); Get(New\_Key);New\_Line;

                                    if Existe(New\_Key,AG,RG) then

                                        Put("Le nom complet de la clé " & Integer'Image(New\_Key) & " est       : ");Put(Full\_Name(New\_Key,RG));Put(".");New\_Line;

                                        Put("La date de naissance de la clé " & Integer'Image(New\_Key) & " est : ");

                                        Afficher\_Date(Birth\_Date(New\_Key,RG));

                                        Put(".");

                                        New\_Line;

                                        Put("Le lieu de naissance de la clé " & Integer'Image(New\_Key) & " est : "); Put(Birth\_Place(New\_Key,RG));Put(".");New\_Line;

                                        if Est\_Nul\_AG(ABR.Fils\_Gauche(Rech\_Noeud\_AG(New\_Key,AG))) and Est\_Nul\_AG(ABR.Fils\_Droit(Rech\_Noeud\_AG(New\_Key,AG))) then

                                            Put\_Line("Cette clé n'a pas de parent connu!");

                                        elsif not Est\_Nul\_AG(ABR.Fils\_Gauche(Rech\_Noeud\_AG(New\_Key,AG))) then

                                            if ABR.NodeValue(ABR.Fils\_Gauche(Rech\_Noeud\_AG(New\_Key,AG)))='M' then

                                                Put("Sa mère a pour clé " & Integer'Image(ABR.Nodekey(ABR.Fils\_Gauche(Rech\_Noeud\_AG(New\_Key,AG)))) & " et s'appelle    : ");

                                                 Put(Full\_Name(ABR.Nodekey(ABR.Fils\_Gauche(Rech\_Noeud\_AG(New\_Key,AG))),RG));Put(".");New\_Line;

                                            elsif ABR.NodeValue(ABR.Fils\_Gauche(Rech\_Noeud\_AG(New\_Key,AG)))='P' then

                                                Put("Son père a pour clé " & Integer'Image(ABR.Nodekey(ABR.Fils\_Gauche(Rech\_Noeud\_AG(New\_Key,AG)))) & " et s'appelle    : ");

                                                 Put(Full\_Name(ABR.Nodekey(ABR.Fils\_Gauche(Rech\_Noeud\_AG(New\_Key,AG))),RG));Put(".");New\_Line;

                                            end if;

                                        end if;

                                        if not Est\_Nul\_AG(ABR.Fils\_Droit(Rech\_Noeud\_AG(New\_Key,AG))) then

                                             if ABR.NodeValue(ABR.Fils\_Droit(Rech\_Noeud\_AG(New\_Key,AG)))='M' then

                                                Put("Sa mère a pour clé " & Integer'Image(ABR.Nodekey(ABR.Fils\_Droit(Rech\_Noeud\_AG(New\_Key,AG)))) & " et s'appelle    : ");

                                                 Put(Full\_Name(ABR.Nodekey(ABR.Fils\_Droit(Rech\_Noeud\_AG(New\_Key,AG))),RG));Put(".");New\_Line;

                                            elsif ABR.NodeValue(ABR.Fils\_Droit(Rech\_Noeud\_AG(New\_Key,AG)))='P' then

                                                Put("Son père a pour clé " & Integer'Image(ABR.Nodekey(ABR.Fils\_Droit(Rech\_Noeud\_AG(New\_Key,AG)))) & " et s'appelle    : ");

                                                 Put(Full\_Name(ABR.Nodekey(ABR.Fils\_Droit(Rech\_Noeud\_AG(New\_Key,AG))),RG));Put(".");New\_Line;

                                            end if;

                                        end if;

                                    end if;

                                    New\_Line;Put\_Line("Retour au menu des affichages dans 3 secondes.."); delay 3.0;

                                    New\_Line;New\_Line;

                                when 2 =>

                                     Put("Saisissez la clé de l'ancêtre dont vous voulez afficher le nom complet : "); Get(New\_Key);New\_Line;

                                     if Existe(New\_Key,AG,RG) then

                                         Put("Le nom complet de la clé " & Integer'Image(New\_Key) & " est : ");Put(Full\_Name(New\_Key,RG));Put(".");New\_Line;

                                     end if;

                                     New\_Line;Put\_Line("Retour au menu des affichages dans 3 secondes.."); delay 3.0;

                                     New\_Line;New\_Line;

                                when 3 =>

                                    Put("Saisissez la clé de l'ancêtre dont vous voulez afficher la date de naissance : "); Get(New\_Key);New\_Line;

                                     if Existe(New\_Key,AG,RG) then

                                         Put("La date de naissance de la clé " & Integer'Image(New\_Key) & " est : ");

                                         Afficher\_Date(Birth\_Date(New\_Key,RG));

                                         Put(".");

                                     end if;

                                     New\_Line;Put\_Line("Retour au menu des affichages dans 3 secondes.."); delay 3.0;

                                     New\_Line;New\_Line;

                                when 4 =>

                                    Put("Saisissez la clé de l'ancêtre dont vous voulez afficher le lieu de naissance : "); Get(New\_Key);New\_Line;

                                     if Existe(New\_Key,AG,RG) then

                                         Put("Le lieu de naissance de la clé " & Integer'Image(New\_Key) & " est : "); Put(Birth\_Place(New\_Key,RG));Put(".");

                                     end if;

                                     New\_Line;Put\_Line("Retour au menu des affichages dans 3 secondes.."); delay 3.0;

                                     New\_Line;New\_Line;

                                when 5 =>

                                    Put("Saisissez la clé de l'ancêtre dont vous voulez afficher les informations des parents : "); Get(New\_Key);New\_Line;

                                    if Existe(New\_Key,AG,RG) then

                                        if Est\_Nul\_AG(ABR.Fils\_Gauche(Rech\_Noeud\_AG(New\_Key,AG))) and Est\_Nul\_AG(ABR.Fils\_Droit(Rech\_Noeud\_AG(New\_Key,AG))) then

                                            Put\_Line("Cette clé n'a pas de parent connu!");

                                        elsif not Est\_Nul\_AG(ABR.Fils\_Gauche(Rech\_Noeud\_AG(New\_Key,AG))) then

                                            if ABR.NodeValue(ABR.Fils\_Gauche(Rech\_Noeud\_AG(New\_Key,AG)))='M' then

                                                Put("Sa mère a pour clé " & Integer'Image(ABR.Nodekey(ABR.Fils\_Gauche(Rech\_Noeud\_AG(New\_Key,AG)))) & " et s'appelle    : ");

                                                 Put(Full\_Name(ABR.Nodekey(ABR.Fils\_Gauche(Rech\_Noeud\_AG(New\_Key,AG))),RG));Put(".");New\_Line;

                                            elsif ABR.NodeValue(ABR.Fils\_Gauche(Rech\_Noeud\_AG(New\_Key,AG)))='P' then

                                                Put("Son père a pour clé " & Integer'Image(ABR.Nodekey(ABR.Fils\_Gauche(Rech\_Noeud\_AG(New\_Key,AG)))) & " et s'appelle    : ");

                                                 Put(Full\_Name(ABR.Nodekey(ABR.Fils\_Gauche(Rech\_Noeud\_AG(New\_Key,AG))),RG));Put(".");New\_Line;

                                            end if;

                                        end if;

                                        if not Est\_Nul\_AG(ABR.Fils\_Droit(Rech\_Noeud\_AG(New\_Key,AG))) then

                                             if ABR.NodeValue(ABR.Fils\_Droit(Rech\_Noeud\_AG(New\_Key,AG)))='M' then

                                                Put("Sa mère a pour clé " & Integer'Image(ABR.Nodekey(ABR.Fils\_Droit(Rech\_Noeud\_AG(New\_Key,AG)))) & " et s'appelle    : ");

                                                 Put(Full\_Name(ABR.Nodekey(ABR.Fils\_Droit(Rech\_Noeud\_AG(New\_Key,AG))),RG));Put(".");New\_Line;

                                            elsif ABR.NodeValue(ABR.Fils\_Droit(Rech\_Noeud\_AG(New\_Key,AG)))='P' then

                                                Put("Son père a pour clé " & Integer'Image(ABR.Nodekey(ABR.Fils\_Droit(Rech\_Noeud\_AG(New\_Key,AG)))) & " et s'appelle    : ");

                                                 Put(Full\_Name(ABR.Nodekey(ABR.Fils\_Droit(Rech\_Noeud\_AG(New\_Key,AG))),RG));Put(".");New\_Line;

                                            end if;

                                        end if;

                                    end if;

                                    New\_Line;Put\_Line("Retour au menu des affichages dans 3 secondes.."); delay 3.0;

                                    New\_Line;New\_Line;

                                when 6 =>

                                    Put("Saisissez la clé de l'ancêtre dont vous voulez afficher le nombre d'ancêtres : "); Get(New\_Key);New\_Line;

                                     if Existe(New\_Key,AG,RG) then

                                         Put("Le nombre d'ancêtres de la clé " & Integer'Image(New\_Key) & " est : " & Integer'Image(Ancestor\_Nbr(New\_Key,AG)) & ".");

                                     end if;

                                     New\_Line;New\_Line;

                                when 7 =>

                                    Put("Saisissez la clé de l'ancêtre dont vous voulez afficher les ancêtres d'une certaine génération : "); Get(New\_Key);New\_Line;

                                    Put("Saisissez la génération : ");Get(g);New\_Line;

                                    if Existe(New\_Key,AG,RG) then

                                         Same\_Gen\_Keys(g,New\_Key,AG);

                                         New\_Line;Put\_Line("Retour au menu des affichages dans 3 secondes.."); delay 3.0;

                                    end if;

                                    New\_Line;New\_Line;

                                when 8 =>

                                    Put("Saisissez la clé de l'ancêtre dont vous voulez afficher les ancêtres d'une certaine génération ou moins: "); Get(New\_Key);New\_Line;

                                    Put("Saisissez la génération : ");Get(g);New\_Line;

                                    if Existe(New\_Key,AG,RG) then

                                         Same\_Gen\_Orless(g,New\_Key,AG);

                                         New\_Line;Put\_Line("Retour au menu des affichages dans 3 secondes.."); delay 3.0;

                                    end if;

                                    New\_Line;New\_Line;

                                when 9 =>

                                    Ensemble\_Orphelins(AG);

                                when 10 =>

                                    Ensemble\_Un\_Parent(AG);

                                when 11 =>

                                    Ensemble\_Deux\_Parents(AG);

                                when 12 =>

                                    if not (Est\_Nul\_AG(AG) and Est\_Vide\_RG(RG)) then

                                        Put\_Line("Affichage de l'arbre complet :");New\_Line;

                                        Print(AG);New\_Line;

                                        Put\_Line("Retour au menu des affichages dans 5 secondes.."); delay 5.0; New\_Line;

                                    else

                                        Put\_Line("L'arbre et le registre sont vides!");New\_Line;New\_Line;

                                    end if;

                                when 13 =>

                                    Put("Saisissez la clé de l'ancêtre à partir duquel vous voulez afficher l'arbre: "); Get(New\_Key);New\_Line;

                                    if Existe(New\_Key,AG,RG) then

                                        Print\_From(New\_Key,AG); New\_Line;

                                        Put\_Line("Retour au menu des affichages dans 5 secondes.."); delay 5.0;

                                    end if;

                                    New\_Line;New\_Line;

                                when 14 =>

                                    MenuPrecedent:=True;

                                    Put\_Line("Retour au menu précédent..."); New\_Line;New\_Line;

                                when others =>

                                    Put\_Line("Saisissez une option valide."); New\_Line;

                            end case;

                        exit when MenuPrecedent;

                        end loop;

                        MenuPrecedent:=False;

                    when 4 =>

                        Quitter:=True;

                    when others =>

                        Put\_Line("Saisissez une option valide."); New\_Line;

                end case;

            exit when Quitter;

            end loop;

        else

            Null;

        end if;

        New\_Line;

        Put\_Line("Vous quitterez le programme dans 2 secondes .."); delay 2.0;New\_Line;

        exception

            when DATA\_ERROR =>

                Skip\_Line;New\_Line;Put\_Line("Vous avez saisi un mauvais type! Redémarrage de l'application dans 3 secondes..");New\_Line; delay 3.0; main;

    end main;