with Piles;

Generic

    TYPE T\_Value is PRIVATE; --Donnée d'une clé.

    Zero: in T\_Value;        --Le "zéro" du type T\_Value. (Sera -34404 ou 'E' pour signifier ERROR quand la fonction devra retourner quelque chose même s'il n'y a rien à retourner)

package Arbre\_Binaire is

        TYPE T\_Node is LIMITED PRIVATE; --Les cellules d'un arbre, i.e. les noeuds, d'où le nom "node" en anglais.

        TYPE T\_Branch is PRIVATE; --Le pointeur qui accèdera aux fils d'un noeud, d'où le nom "branche".

        package Piles\_Cle is

            new Piles (T\_Element => Integer);

        use Piles\_Cle; --Module de piles qui sera très utile plus tard.

                              --EXCEPTIONS--

    Cle\_Presente\_Exception : Exception; -- une clé est déjà présente dans un ABR

    Cle\_Absente\_Exception  : Exception; -- une clé est absente d'un ABR

    Arbre\_Vide             : Exception;

                     --FONCTIONS/PROCEDURES SECONDAIRES--

        --Procédure obligatoire pour instancier le module des piles.

    procedure Afficher\_Entier(*Element*: in Integer);

                                --TESTS--

        --Initialiser un arbre vide.

    procedure Initialiser\_Vide(*Arbre*: out T\_Branch);

        --Initialiser un arbre avec  la clé de sa racine éventuelle.

    procedure Initialiser (*Cle*: in Integer; *Arbre* : out T\_Branch);

        -- Est-ce que le noeud s'agit d'une feuille? Est-ce que l'abre est vide?

    function Est\_Nul (*Arbre* : in T\_Branch) return Boolean;

        --Savoir si une clé est la racine ou bien le fils droit ou le fils gauche de son ancêtre. (retourne 'R' pour racine, 'D' pour fils droit et 'G' pour fils gauche)

    function Gauche\_ou\_Droite(*Cle*:in Integer;*Arbre*: in T\_Branch) return Character;

                        --FONCTIONS ELEMENTAIRES--

        --Retourner la clé du noeud.

    function Nodekey(*Arbre*: in T\_Branch) return Integer;

        --Retourner la valeur du noeud.

    function NodeValue(*Arbre*:in T\_Branch) return T\_Value;

        --Retourner le fils droit du noeud.

    function Fils\_Droit(*Arbre*: in T\_Branch) return T\_Branch;

        --Retourner le fils gauche du noeud.

    function Fils\_Gauche(Arbre: in T\_Branch)return T\_Branch;

        --Multiplier toutes les clés de l'arbre par 10.

    procedure Multiplier\_10(*Arbre*: in out T\_Branch);

        --Retourner la profondeur d'un arbre.

    function Depth(*Arbre*: in T\_Branch) return Integer;

        --Affecter un noeud par un autre

    procedure Affecter\_Arbre(*Arbre1*: in out T\_Branch; *Arbre2*: in T\_Branch);

                    --FONCTIONS/PROCEDURES GENERATIONNELLES-

        --Récupérer la génération d'un noeud par rapport à la racine.

    function Gen(*Cle*: in Integer; *Arbre*:in T\_Branch) return Integer;

        --Récupérer le nombre de fils d'un noeud.

    function Nbr\_Fils\_Noeud(*Cle*: in Integer; *Arbre*: in T\_Branch) return Integer;

        --Récupérer l'ensemble des fils d'un noeud

    function Ensemble\_Fils\_Noeud(*Cle*: IN Integer;*Arbre*: in T\_Branch) return Piles\_Cle.T\_Pile;

        --Récupérer le nombre de noeuds de génération g.

    function Nbr\_Meme\_Generation(*g*: in integer; *Arbre*: in T\_Branch) return Integer;

        --Afficher l'ensemble des noeuds de génération g par rapport à un noeud.

    procedure Ensemble\_Meme\_Generation(*g,Cle*:in Integer; *Arbre*: in T\_Branch);

        --Afficher l'ensemble des noeuds de génération g ou moins par rapport à un noeud.

    procedure Ensemble\_n\_Generation(*g,Cle*:in Integer; *Arbre*: in T\_Branch);

        --Récupérer l'ensemble des noeuds ayant un seul fils.

    function Ensemble\_Un\_Fils(*Arbre*:in T\_Branch) return Piles\_Cle.T\_Pile;

        --Récupérer l'ensemble des noeuds ayant deux fils.

    function Ensemble\_Deux\_Fils(*Arbre*:in T\_Branch) return Piles\_Cle.T\_Pile;

        --Récupérer l'ensemble des feuilles d'un arbre.

    function Ensemble\_Feuilles(*Arbre*:in T\_Branch) return Piles\_Cle.T\_Pile;

                    --FONCTIONS/PROCEDURES DE RECHERCHE--

        --Récupérer la cellule d'un noeud à partir de sa clé.

    function Rech\_Noeud(*Cle*: in Integer; *Arbre*: in T\_Branch) return T\_Branch;

        --Récupérer la cellule de l'ancêtre d'un noeud à partir de sa clé.

    function Rech\_Ancetre(*Cle*: in Integer; *Arbre*: in T\_Branch) return T\_Branch;

        --Affecter à Noeud la cellule de clé Cle dans l'arbre (i.e., affecter le resultat de Rech\_Noeud(Cle,Arbre) à Noeud).

    procedure Affecter\_Rech\_Noeud(*Cle*: in Integer;*Arbre*: in T\_Branch; *Noeud*: in out T\_Branch);

        --Récuperer la donnée associée à une clé dans un arbre.

    function Donnee\_Noeud(*Cle* : in Integer ; *Arbre* : in T\_Branch) return T\_Value;

        --Récupérer la clé associée à une clé dans un arbre.

    function Cle\_Noeud(*Cle*:in Integer; *Arbre*:in T\_Branch) return Integer;

                                --AJOUT--

        --Récupérer l'intervalle (piles) de valeurs permises que peut prendre le fils éventuel d'un noeud en testant avec une clé quelconque.

        --par convention, la pile sera de la forme [max,0,-181199] pour représenter l'intervalle ]-INFINI,max], [-181199,0,min] pour représenter l'intervalle [min,+INFINI[ et [max,min] pour représenter l'intervalle [min,max].

    function NewKeyInterval(*Cle,Parent*: Integer; *Arbre*:T\_Branch) return Piles\_Cle.T\_Pile;

        --Insérer un nouvel élément (clé + donnée) dans l'arbre binaire de façon automatique.

    procedure Inserer(*Cle*: in Integer; *Donnee*: in T\_Value; *Arbre*: in out T\_Branch);

        --Insérer un nouvel élément (clé + donnée) sous un noeud précis.

    procedure Ajouter2(*Cle\_Nouveau\_Noeud*: in Integer; *Donnee\_Nouveau\_Noeud*: in T\_Value; *Cle\_Noeud\_Parent*:in integer; *Arbre*: in out T\_Branch);

                             --MODIFICATION--

        --Modifier la clé de la racine ou du noeud poussé en argument.

    procedure Modifier\_Cle\_Racine(*NewCle*:in Integer;*Arbre*: in out T\_Branch);

        --Modifier la clé d'un noeud.

    procedure Modifier\_Cle(*Cle,NewCle*: in Integer; *Arbre*: in out T\_Branch);

        --Modifier la donnée d'un noeud.

    procedure Modifier\_Donnee(*Cle*: in Integer; *NewDonnee*: in T\_Value; *Arbre*: in out T\_Branch);

                              --SUPPRESSION--

        -- Supprimer les fils d'un noeud de l'arbre.

    procedure Supprimer\_Fils (*Arbre*: in out T\_Branch);

        -- Supprimer un noeud de l'arbre ainsi que ses fils.

    procedure Supprimer\_Cle\_ET\_Fils(*Cle*: in Integer; *Arbre*: in out T\_Branch);

        -- Détruire tout un arbre ou un noeud et tous ses fils.

    procedure Detruire(*Arbre*: in out T\_Branch);

                              --AFFICHAGE--

    generic

        with procedure Afficher\_Donnee (*Donnee*: in T\_Value);

    procedure Afficher\_ABR (*Arbre* : in T\_Branch); --Afficher l'arbre complet.

    generic

        with procedure Afficher\_Donnee (*Donnee*: in T\_Value);

    procedure Afficher\_APartir(*Cle*:in Integer; *Arbre*: in T\_Branch); --Afficher un arbre à partir d'une clé.

PRIVATE

    TYPE T\_Branch is ACCESS T\_Node; --Pointeur qui accèdera aux noeuds.

    TYPE T\_Node is

        RECORD

            Cle: Integer;           --La clé d'un noeud.

            Donnee: T\_Value;        --La donnée d'une clé.

            FilsG: T\_Branch;        --Pointeur qui accède au fils gauche d'un noeud.

            FilsD: T\_Branch;        --Pointeur qui accède au fils droit d'un noeud.

        end RECORD;

end Arbre\_Binaire;