```
with Ada.Unchecked_Deallocation;
with Ada.Text_IO;               use Ada.Text_IO;

package body Piles is

    procedure Free is
        new Ada.Unchecked_Deallocation (T_Cellule, T_Pile);


                        --TESTS--

    procedure Initialiser (Pile : out T_Pile) is
    begin
        Pile := Null;
    end Initialiser;

    function Est_Vide (Pile : in T_Pile) return Boolean is
    begin
        return Pile = Null;
    end Est_Vide;

    function Existe_Pile(Element: in T_Element; Pile: in T_Pile) return Boolea
n is
        Parcours:T_Pile:=Pile;
        begin
            while Parcours/=Null loop
                if Sommet(Parcours)=Element then
                    return True;
                end if;
                Parcours:=Parcours.all.Suivant;
            end loop;
            return False;
        end Existe_Pile;


                --FONCTIONS ELEMENTAIRES--

    function Sommet (Pile : in T_Pile) return T_Element is
    begin
        if Pile/=Null then
            return Pile.all.Element;
        else
            raise Pile_Vide;
        end if;
    end Sommet;

    function Next_Pile(Pile: in T_Pile) return T_Pile is
        begin
            if Pile=Null then
```

```
                return Null;
            else
                return Pile.all.Suivant;
            end if;
        end Next_Pile;

    function Size_Pile(Pile: in T_Pile) return Integer is
        Parcours:T_Pile;
        i:Integer:=0;
        begin
            if Est_Vide(Pile) then
                return 0;
            else
                Parcours:=Pile;
                while not Est_Vide(Parcours) loop
                    i:=i+1;
                    Parcours:=Parcours.all.Suivant;
                end loop;
                return i;
            end if;
        end Size_Pile;


                  --AJOUT/SUPPRESSION--

    procedure Empiler (Pile : in out T_Pile; Element : in T_Element) is
        Nouvelle_Cellule: T_Pile;
    begin
        Nouvelle_Cellule := new T_Cellule;
        Nouvelle_Cellule.all.Element := Element; --
Nouvelle cellule contenant l'élément
        Nouvelle_Cellule.all.Suivant := Pile; --
Rattacher le "bout" de cette cellule avec le reste de la pile
        Pile := Nouvelle_Cellule; --
La pile contient maintenant l'élément empilé ainsi que le reste de la pile
    end Empiler;

    procedure Depiler (Pile : in out T_Pile) is
        A_Detruire : T_Pile;
    begin
        A_Detruire := Pile;
        Pile := Pile.all.Suivant; --
Le sommet de la pile est maintenant le second élément
        Free (A_Detruire); --Detruction de l'ancien sommet
    end Depiler;

    procedure Supprimer_Element(Element: in T_Element; Pile: in out T_Pile) is
        Parcours,A_Supprimer: T_Pile;
```

```
        begin
            if Pile=Null then
                Null;
            elsif not Existe_Pile(Element, Pile) then
                Null;
            else
                if Sommet(Pile)=Element then
                    Depiler(Pile);
                else
                    Parcours:= Pile;
                    while Parcours.all.Suivant/=Null and then Sommet(Parcours.
all.Suivant)/=Element 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 Supprimer_Element;

    procedure Detruire (P: in out T_Pile) is
    begin
        if P /= Null then
            Detruire (P.all.Suivant);
            Free (P); --
Detruction de la pile à partir du premier élément empilé jusqu'au sommet
        else
            Null;
        end if;
    end Detruire;

    procedure Affecter_Pile(P:in out T_Pile;P2: in T_Pile) is
        begin
            P:=P2;
        end Affecter_Pile;

                    --AFFICHAGE--

    procedure Afficher_Pile (Pile : in T_Pile) is
        procedure Afficher_Elements (Pile : in T_Pile) is
        begin
            if Pile = Null then
                Null;
            elsif Pile.all.Suivant = Null then
                Put (" ");
                Afficher_Element (Pile.all.Element);
            else
```

```
                Afficher_Elements (Pile.all.Suivant);
                Put (", ");
                Afficher_Element (Pile.all.Element);
            end if;
        end Afficher_Elements;
    begin
        Afficher_Elements (Pile);
        Put (" ."); --La pile sera affichée comme W, X, Y, Z .
    end Afficher_Pile;


end Piles;
```