Contrôles Sémantiques

1. Addition d'un entier et d'un flottant

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure Semantic_1 is
  X : Integer :=1 ;
  Y : Float := 3.14;
  Z : Integer := X + Y; -- Erreur : types incompatibles
begin
end Semantic_1;
```

2. Mauvais identifiant de fonction

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure Semantic_2 is
  function calcul1(val1 : integer; val2 : integer; val3 : integer) return
integer is
    result : integer;
    begin
      result := val1 + val2 * val3;
    return result;
  end fonction1 ; -- Erreur : fonction1 au lieu de calcul1
begin
end Semantic_2;
```

3. Variable utilisée avant sa déclaration

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure Semantic_3 is
   X : Integer := Y; -- Erreur : Y n'est pas encore déclaré
   Y : Integer := 42;
begin
end Semantic_3;
```

4. Tentative de modifier un paramètre en mode "in"

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure Semantic_4 is
   procedure My_Procedure(Item : in Integer) is
   begin
      Item := 42; -- Erreur : Tentative de modifier un paramètre en mode
"in"
   end My_Procedure;
begin
   My_Procedure(10);
end Semantic_4;
```

5. Mauvais type d'affectation pour un membre d'une structure

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure Semantic_5 is
  type point is record
    x : integer ;
    y : integer ;
  end record;
begin
  p point;
  p.x := 8.4; -- Erreur : 8.4 n'est pas un entier
end Semantic_5;
```

6. Accès à un membre inexistant d'une structure

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure Semantic_6 is
  type point is record
    x : integer ;
    y : integer ;
  end record;
begin
  p point;
  p.z := 20; -- Erreur : z n'est pas un membre du type point
end Semantic_6;
```

7. Utilisation d'autre chose qu'un entier pour le maximum d'une boucle for

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure Semantic_7 is
   var : Integer := 0;
begin
   for i in 1..5.4 loop -- Erreur : 5.4 n'est pas un entier
      var := var * 2;
   end loop;
end Semantic_7;
```

8. Nombre d'arguments lors de l'appel d'une fonction

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure Semantic_8 is
  function calcul1(val1 : integer; val2 : integer; val3 : integer) return
integer is
    result : integer;
    begin
      result := val1 + val2 * val3;
    return result;
  end calcul1 ;
  var : Integer := calcul1(10,5);  -- Erreur : 2 paramètres au lieu de 3
begin
end Semantic_8;
```

9. Appel d'une fonction inexistante

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure Semantic_9 is
  function calcul1(val1 : integer; val2 : integer; val3 : integer) return
integer is
    result : integer;
    begin
      result := val1 + val2 * val3;
    return result;
  end calcul1 ;
```

```
  var : Integer := calcul2(10,5,9);  -- Erreur : calcul2 n'est pas définie
begin
end Semantic_9;
```

10. Variable accédée alors qu'elle n'a pas été initialisée

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure Semantic_10 is
   var : Integer := 0;
begin
   variable := var * 2;   -- Erreur : variable n'a pas été intialisée
end Semantic_10;
```

11. Bon nombre de params lors d'un appel, mais mauvais types

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure Semantic_11 is
  function calcul1(val1 : integer; val2 : integer; val3 : integer) return
integer is
    result : integer;
    begin
      result := val1 + val2 * val3;
    return result;
  end calcul1 ;
  var : Integer := calcul1("10",5);  -- Erreur : Integer attendu en param
1 mais String fourni
end Semantic_11;
```

12. Absence de "return" pour une fonction

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure Semantic_12 is
  function calcul1(val1 : integer; val2 : integer; val3 : integer) return
integer is
    result : integer;
    begin
      result := val1 + val2 * val3;
    -- Erreur: Il manque le return de la valeur
  end calcul1 ;
```

```
  var:integer;
begin
   Put("Hello, World!");
end Semantic_12;
```

13. Accès à une variable hors du bloc

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure Semantique_13 is
   procedure Test is
       A : Integer := 10;
   begin
       Put("test");
   end Test;

   function Ma_Fonction return Integer is
   begin
      Put("test");
      return A; --Erreur: Tentative d'accéder à A à l'extérieur de son
bloc
   end Ma_Fonction;
 begin
    Put("Hello World!");
 end Semantique_13;
```

14. Division statique par 0 (ou 0.0)

```
with Ada.Text_IO;
use Ada.Text_IO;
procedure Semantic_14 is
   Resultat : Integer;
begin
   Resultat := 1 / 0; -- Erreur: division par 0
end Semantic_14;
```

15. Accès à une variable hors du bloc #2

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

procedure Semantic_15 is
```

```
    function calcul1(val1 : integer; val2 : integer; val3 : integer)
return integer is
    N1:integer;
    result1: integer;
        function calcul2(val1 : integer) return integer is
            N2 : integer;
            result2:integer;
            begin
                N2 :=4;
                result2 := val1 * val1;
            return result2;
        end calcul2 ;
        begin
            N1 := N2; -- Pas accès à la variable N2
            result1 := val1 + val2 * val3;
        return result1;
    end calcul1 ;
x : integer :=1;
y : integer :=4;
z : integer :=2;
var: integer;
begin
    var:= calcul1(x,y,z);

end Semantic_15;
```

16. Le résultat d'une opération ne correspond pas au type de la variable

```
Ada.Text_IO;
use Ada.Text_IO;
procedure Semantic_16 is
  X : Integer :=1 ;
  Y : Integer := 2;
  Z : String := X + Y; -- Erreur : + renvoie un integer alors que Z attend
un string
begin
end Semantic_16;
```

17. Tentative d'accès à une fonction non présente dans le contexte d'appel

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

procedure Semantic_17 is
    function calcul1(val1 : integer; val2 : integer; val3 : integer)
return integer is
    N1:integer;
    result1: integer;
        function calcul2(val1 : integer) return integer is
            N2 : integer;
            result2:integer;
            begin
                N2 :=4;
                result2 := val1 * val1;
            return result2;
        end calcul2 ;
        begin
            N1 := calcul2(val1);
            result1 := calcul3(val2); -- Pas accès à la fonction calcul3
        return result1;
    end calcul1 ;
    function calcul3(val1 : integer) return integer is
        N1_bis : integer;
        result3:integer;
        begin
            N1_bis :=6;
            result3 := val1 + val1;
        return result3;
    end calcul3 ;
x : integer :=1;
y : integer :=4;
z : integer :=2;
var: integer;
begin
    var:= calcul1(x,y,z);

end Semantic_17;
```

18. Tentative d'accès à une variable non présente dans le contexte d'appel

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

```
procedure Semantic_18 is
    function calcul1(val1 : integer; val2 : integer; val3 : integer)
return integer is
    N1:integer;
    result1: integer;
        function calcul2(val1 : integer) return integer is
            N2 : integer;
            result2:integer;
            begin
                N2 :=4;
                result2 := val1 * val1;
            return result2;
        end calcul2 ;
        begin
            N1 := 8;
            result1 := calcul1(x); --Problème accès à la variable x
        return result1;
    end calcul1 ;
x : integer :=1;
y : integer :=4;
z : integer :=2;
var: integer;
begin
    var:= calcul1(x,y,z);

end Semantic_18;
```