```
Procedure Insertion_AVL (x :entier, var A :ptr (nœud_avl))

Var
T, P :ptr(nœud_avl) ;
Pile_parent :Pile//vers ptr(nœud_avl)
Debut
CreerPile(pile_Parent)
Si (A !=nil) alors
  Trouv :=faux
  Insert :=faux
  Tantque (A !=nil) et (!trouv) et (!Insert)      faire
  Dtq
  | Si (x=info(A)) alors trouv :=vrai
  | Sinon
  |
  |   Si x>info(A) alors
  |   | Si (FD(A)=nil) alors
  |   |       P:=CréerNoeud(x) ;
  |   |       Aff_bal(P,0) ;
  |   |       Aff_FD(A,P) ;
  |   |       Aff_bal(A, balance(A)-1) //car insertion à droite de A Insert
  |   |      :=vrai ;
  |   | Sinon
  |   |       Empiler (Pile_Parent,A) //en empile tous les pères A
  |   |       :=FD(A) ;
  |   Sinon
  |   | Si (FG(A)=nil) alors
  |   |     P:=CréerNoeud(x) ;
  |   |     Aff_bal(P,0) ;
  |   |     Aff_FG(A,P) ;
  |   |     Aff_bal(A, balance(A)+1) //car insertion à gauche de A
  |   |     Insert=vrai ;
  |   | Sinon
  |   |       Empiler (Pile_Parent,A) //en empile tous les pères
  |   |       A :=FG(A) ;
  | Ftq
Sinon //A=nil
 A=créer_noeud(x ) ;
 Aff_bal(A,0);
Fsi
//chercher le premier antécédent  dans la pile_parent Si
(insert) alors
Fin=faux
```

```
Tantque !pile_vide(pile_parent) et !Fin
Dtq
  Dépiler (Pile_parent, T)
  Si A=FD(T) alors aff_Bal(T, balance(T)-1)
                Sinon aff_Bal (T,balance (T)+1)
  Si (balance(T) <=-2) ou (Balance>=2) ou balance (T)=0 alors
            Fin=vrai ;
            Si balance(T) !=0  alors
              Dépiler(pile_parent, père)//pour lier T à son père car T change
              Equilibrer (T, père) ;
            fsi
  Sinon
  A :=T ; //continuer la recherche de l'antécedent
Ftq
fsi
Fin
```

Remarque : si balance d'un nœud=0, on arrête la recherche et on fait rien car l'arbre est équilibré (la balance des nœuds se trouvant au dessus ne changera pas).

Procédure Equilibrer (var A: ptr (nœud_Avl), var pereA: ptr (nœud_Avl))

```
Var
B,Q : ptr(nœud-Avl) ;
Debut
Si (balance(A)=-2) alors //deux cas RotG ou DRotG
 si (balance (FD(A)<=0) alors //on fait une simple rotation gauche de A B
 :=ROTG(A);
 sinon    // Double Rotation gauche de A=RotD(FD(A))+RotG(A)
 Q:=ROTD(FD(A));
   Aff-FD(A,Q)
   B :=ROTG(A);

Sinon //RotD ou bien DRotD
Si (balance(A)=2) alors
Si balance (FG(A)>= 0) alors
    B :=ROTD(A);
  sinon
   Q:=ROTG(FG(A));
   Aff-FG(A,Q)
   B:=ROTD(A);
Fsi
Si (pèreA !=nil)  alors
Si  A =FD(pereA) alors Aff-FD(pereA,B)
                    Sinon Aff-FG(pereA,B);
Fsi

Fin
```

Version récursive de l'insertion

```
Insertion (x :Entier, A :ptr(nœud_Avl, h :entier)
Var : T :ptr(nœud-AVL), h1 : entier / / h, h1 désignent la variation de hauteur (1,-1,0)
Si (A=nil) alors
  A:= créerNoeud(x)
  Aff-bal(A,0);
  H:=1
Sinon
  Si (x=info(A) alors h :=0
  Sinon
   Si (x>info(A) alors / /insertion comme FD(A)
   T := FD(A)
     Inserer(x,T,h1)
    Aff_FD(A,T) / /affecter le nouveau T comme fils droit de A, il peut être inchangé ou celui créé
                    ou alors un autre suite à une rotation
     h :=-h1  / /h1 peut être au retour = 1,-1,0
   sinon
     T := FG(A)
      Inserer(x,T,h1)
   Aff_FG(A,T)
   h:=h1  / /pareil
    fsi
fsi
 / /mis à jour de la balance
Si(h !=0) alors     / /si la variation est de 0, les balances des nœuds supérieurs restent
                    inchangées
 Aff-bal(A, balance(A)+h)
 Si balance(A)=2 ou balance A=-2 alors Equilibrer (A)  //A est en E/S
 Si balance (A)=0 alors h :=0 / /on changera plus les balances
                  Sinon h :=1
FIN
```

Remarque :

-Si la balance d'un nœud passe à 0 les nœuds ascendants ne seront pas affectés (leurs balances restent inchangées), exemple le nœud 40 était à 1 et reste à 1 après insertion du 35 car balance du nœud 35 est passée de 1 à 0

- La fonction Equilibrer reste la même mais ne prend pas en compte pereA (le chainage est pris en compte dans l'algorithme recursif (Aff-fD(A,T) et (Aff-fg(A,T))