Les arbres :

## Le nombre de nœuds :

**Type** nœud=**structure**

Info : **typeqlq**

FG, FD : **ptr(nœud)**

**Fin.**

**Var** Arbre : **ptr (nœud)**

### Récursif :

**Fonction**Nb\_Noeuds (R : **ptr(nœud)**) : **entier**

**Debut**

1. **SI** (R=NIL) alors Nb\_Noeuds🡨0
2. **SINON**
   1. Nb\_Nœuds🡨1+Nb\_Nœuds(FG(R))+Nb\_Nœuds(FD(R))
3. **FSI**

**FIN.**

### Itératif :

**Fonction** Nb\_Nœuds(R : Ptr(Nœud)) : entier

**Var** Q : Ptr(Nœud)

P : Ptr(Pile)

Nb : entier

**Debut**

1. **TQ** (R<>NIL) ou (Pile\_Vide(P) faire
   1. **TQ** (R <> NIL) faire
      1. Nb🡨Nb+1
      2. EMPILER(P,R)
      3. R🡨FG(R)
   2. **FTQ**
   3. DEPILER(P,R)
   4. P🡨FD(R)
2. **FTQ**
3. Nb\_Nœuds🡨Nb

**Fin.**

## Nombre de feuilles :

### Récursif :

**Fonction** Nb\_Feuilles(R : ptr(Nœud)) : entier

**Debut**

1. **SI** (R=NIL) Nb\_Feuilles🡨0
2. **SINON**
   1. **SI** (FG(R)=NIL) et (FD(R)=NIL) alors Nb\_feuilles🡨1
   2. **SINON**
   3. Nb\_feuilles🡨Nb\_feuilles(FG(R))+Nb\_feuilles(FD(R))
   4. **FSI**
3. **FSI**

**Fin.**

### Itératif :

**Fonction** Nb\_Feuilles(R : ptr(nœud) : entier

**Var** Nb : entier

**Debut**

1. Créer\_pile(P)
2. Nb🡨0
3. **TQ** (R<>NIL) ou (>Pile-vide(p)) faire
   1. **TQ** (R<> NIL) faire
      1. **SI** (FD(R)=NIL) et (FG(R)=NIL) alors Nb🡨Nb+1
      2. EMPILER(P,R)
      3. R🡨FG(R)
   2. **FTQ**
   3. DEPILER(P,R)
   4. R🡨FD(R)
4. **FTQ**
5. Nb-feuilles🡨Nb

**Fin.**

## Somme des contenus de tous les nœuds :

### Récursif :

**Fonction** Somme\_Nœuds(R : ptr(Nœud)) : entier

**Debut**

1. **Si** (R=NIL) Somme\_Noeuds🡨0
2. **SINON**
   1. Somme\_noeuds🡨Info(R)+Somme\_Noeuds(FG(R))+Somme\_Noeuds(FD(R))
3. **FSI**

**Fin.**

### Itératif :

**Fonction** Somme\_noeuds(R : Ptr(Nœud)) : entier

**Var** Som : entier

**Debut**

1. Créer\_Pile(P)
2. Som🡨0
3. **TQ** (R<>NIL) ou (>Pile-Vide(P)) faire
   1. **TQ** (R<>Nil) Faire
      1. Som🡨Som+Info(R)
      2. EMPILER(P,R)
      3. R🡨FG(R)
   2. **FTQ**
   3. DEPILER(P,R)
   4. R🡨FD(R)
4. **FTQ**
5. Somme\_Noeuds🡨Som

**Fin.**

## La profondeur :

### Récursif :

**Fonction** Profondeur(R : Ptr(Nœud)) : entier

**Debut**

1. **Si** (R=Nil) Profondeur🡨-1
2. **SINON**
   1. Profondeur🡨1+Max(Profondeur(FG(R),Profondeur(FD(R)))
3. **FSI**

**Fin.**

### Itératif :

**Fonction** Profondeur (R : ptr(Nœud)) : entier

**Var** Prof, max : entier

**Debut**

1. Créer\_Pile(P)
2. Prof🡨0
3. Max🡨-1
4. **TQ** (R<>Nil) ou(>Pile-vide(p)) faire
   1. Prof🡨prof-1
   2. **TQ** (R<>Nil) faire
      1. Prof🡨Prof+1
      2. EMPIlER(P,R)
      3. R🡨FG(R)
   3. **FTQ**
   4. **Si (**Prof>Max) alors max🡨prof
   5. DEPILER(P,R)
   6. R🡨FD(R)
5. **FTQ**
6. Profondeur🡨Max

**Fin.**

## Arbre strictement binaire :

### Récursif :

**Fonction** bin\_Strict(R : ptr(nœud)) : booléen

**Debut**

1. **Si** (R=Nil) alorsBin\_Strict🡨Vrai
2. **Sinon**
   1. **Si** [(FG(R)=Nil)et(FD(R)<>Nil)] ou [(FG(R)<>Nil) et (FD(R)=Nil) alorsBin\_Strict🡨faux
   2. **Sinon** Bin\_Strict🡨[Bin\_strict(FG(R))]et[Bin\_strict[FD(R)]
   3. **FSI**
3. **FSI**

**Fin.**

## Complet :

### //elle n’est pas récursive car elle appelle une autre fonction//

**Fonction** Complet (R : Ptr(nœud)) : Booléen

**Début**

1. **Si** (R=Nil) alors Complet🡨vrai
2. **Sinon** 
   1. **Si** (Bin\_Strict(R) et memelvl(R)) alors complet🡨vrai
   2. **Sinon** Complet🡨faux
   3. **FSI**
3. **FSI**

**Fin.**

**Fonction** Arb\_Complet(R :ptr(Nœud)) :Booléen

**Debut**

1. **Si** (R=Nil) alors Arb\_Complet🡨vrai
2. **Sinon**
   1. **Si** [(Profondeur(FG(R))<>Profondeur(FD(R))] ou [(Strict\_bin(FG(R)) et (>Strict\_bin(FD(R)) ou (>strict\_bin(FG(R)) et (Strict\_bin(FD(R))] alors
      1. Arb\_Complet🡨Faux
   2. **Sinon**
      1. Arb\_complet🡨(arb\_complet(FG(R)) et (Arb\_Complet(FD(R))
   3. **FSI**
3. **FSI**

**Fin.**

# Trouver les doubles dans une suite de n nombres :

# Algorithme de parcours :

## Pré ordre :

### Récursif :

**Procedure**Preordre(R :Ptr(nœud))

**Debut**

1. **Si** (R<>Nil) alors
   1. Ecrire(Info(R))
   2. Preordre(FG(R))
   3. Preordre(FD(R))
2. **FSI**

**Fin.**

### Itératif :

**Procedure**Preordre(R :ptr(Nœud))

**Debut**

1. Créer\_pile(p)
2. **TQ** (R<>Nil) ou (>Pile\_Vide(p))
   1. **TQ** (R<>Nil)
      1. Ecrire(info(R))0
      2. **Si** (FD(R)<>Nil)
         1. Empiler(p,FD(R))
      3. **FSI**
      4. R🡨FG(R)
   2. **FTQ**
   3. Depiler(P,R)
3. **FTQ**

**Fin.**

## Innordre :

### Récursif :

**Procedure**Inordre(R :ptr(Nœud))

**Debut**

1. **Si** (R<>Nil) alors
   1. Inordre(Fg(R))
   2. Ecrire(Info(n))
   3. Inordre(FD(R))
2. **FSI**

**Fin.**

### Itératif :

**Procedure**Inordre(R :ptr(Nœud))

**Debut**

1. Créer\_pile(p)
2. **TQ** (R<>Nil) ou (>Pile\_Vide(p))
   1. **TQ** (R<>Nil) faire
      1. Empiler(p,R)
      2. R🡨FD(R)
   2. **FTQ**
   3. Depiler(R,p)
   4. Ecrire(Info(R))
   5. R🡨FD(R)
3. **FTQ**

**Fin.**

## Post ordre :

### Récursif :

**Procedure**Postordre(R :ptr(Nœud))

**Debut**

1. **Si** (R<>Nil) alors
   1. Postordre(Fg(R))
   2. Ecrire(Info(n))
   3. Postordre(FD(R))
2. **FSI**

**Fin.**

### Itératif :

**Type** Sauv=**Structure**

Nœud : ptr(nœud)

Indice : booléen

**Fin.**

**Procedure**Post\_Ordre(R :ptr(nœud))

**Debut**

1. **TQ** (R<>Nil) ou (>PileVide(P) faire
   1. **TQ** (R<>Nil) faire
      1. Sauv.Nœud🡨R
      2. Sauv.indice🡨Vrai
      3. Empiler(P,Sauv)
      4. R🡨FG(R)
   2. **FTQ**
   3. Depiler(P,Sauv)
   4. R🡨Sauv.Nœud
   5. **Si** (Sauv.indice) alors
      1. Sauv.indice🡨faux
      2. Empiler(p,sauv)
      3. R🡨FD(R)
   6. **Sinon**
      1. Ecrire(Info(R))
      2. R🡨 Nil
   7. **FSI**
2. **FTQ**

**Fin.**

# Recherche, Insertion et Suppression :

## Recherche :

**Procedure**Recherche (R : ptr(nœud), var Q :ptr(nœud), var père : ptr(nœud), var Trouv : Booléen)

**Var** P : ptr(nœud)

**Debut**

Trouv🡨faux

P🡨R

Père🡨NIL

Q🡨NIL

**TQ** (R<>NIL) et (>Trouv)

1. **SI** (Info(p)=v) alors
   1. Trouv🡨vrai
   2. Q🡨P
2. **Sinon**
   1. père🡨P
   2. **Si** (Info(P)>v) alors
      1. P🡨FG(P)
   3. **Sinon**
      1. P🡨FD(P)
   4. **FSI**
3. **FSI**

**FTQ**

**Fin.**

## Insertion :

**Fonction** Insertion (R : Ptr(nœud), V : entier) : Ptr(Nœud)

**Var** P, Père : Ptr (nœud)

Trouver : Booléen

**Debut**

1. Recherche(R, P, Père, Trouver, v)
2. **Si** (Trouver=vrai) ou (P<>NIL) alors Ecrire (‘Insertion impossible’)
3. **Sinon** 
   1. Créer\_ Nœud (P)
   2. Aff\_ Info (P,v)
   3. **Si** (Père=NIL) alors R🡨P//L’arbre n’existe pas
   4. **Sinon** 
      1. **Si** (V<Info(Père)) alors Aff\_ FG (Père, P)
      2. **Sinon** Aff\_ FD (Père, P)
   5. **FSI**
4. **FSI**

**Fin.**

## Suppression :

**Procédure** Supprimer (R : Ptr(nœud), Q :Ptr(Nœud)

**Debut**

1. Recherche (R, P, Per, Trouv, Val)
2. **Si** (P=NIL) alors Ecrire (‘Suppression Impossible’)
3. **Sinon** 
   1. **Si** (FG(P)=NIL) alors
      1. **Si** (FD(P)=NIL) alors //Cas n°1
         1. Chaînage (R, Père, P, Val, NIL)
      2. **Sinon** //Cas n°2
         1. Chaînage (R, Père, P, Val, FD(P))
      3. **FSI**
   2. **Sinon**
      1. **Si** (FD(P)=NIL) alors //Cas n°3
         1. Chaînage (R, Père, P, Val, FG(P))
      2. **Sinon** //Cas n°4
         1. Cas\_Quatre (P)
      3. **FSI**
   3. **FSI**
4. **FSI**

**Fin.**

**Procédure** Chaînage (Var R :ptr, Père :ptr, P :ptr, val : entier, Q :ptr)

**Debut**

1. **Si** (Père=NIL) alors R🡨Q
2. **Sinon**
   1. **Si** (Val>Info(Père)) alors Aff\_FD (Père, Q)
   2. **Sinon** Aff\_FG (Père, Q)
3. **FSI**
4. Liberer (P)

**Fin.**

**Procédure** Plus\_ Petit (Var PElt :ptr, Var Elt :entier, Var Per\_elt : ptr)//Utilisée pour le quatrième cas

**Début**

1. **TQ** (FG (Pelt) <> NIL) faire
   1. Père\_elt🡨PElt
   2. PElt🡨FG (PElt)
2. **FTQ**
3. Elt🡨Info(PElt)

**Fin.**

**Procédure** Cas\_Quatre (P : ptr)

**Var** Q, PèreQ : ptr

Val : entier

**Début**

1. Q🡨FD(P)
2. PèreQ🡨P
3. Plus\_Petit(Q, Val, PèreQ)
4. Aff\_Info (P, Val)
5. **Si** (PèreQ=P) alors Aff\_FD (PèreQ,FD(Q))
6. **Sinon** Aff\_FG (PèreQ, FD(Q))
7. Libérer (Q)

**Fin.**