Implémentation des arborescences binaires

le header : arborescences binaires

```
typedef int Valeur ;
struct Sommet ;
```

le header : arborescences binaires

```
typedef int Valeur ;
struct Sommet ;
typedef Sommet* AB ;
```

le header : arborescences binaires

```
typedef int Valeur;
struct Sommet;
typedef Sommet* AB;
struct Sommet {

};
```

le header : arborescences binaires

```
typedef int Valeur;
struct Sommet;
typedef Sommet* AB;
struct Sommet {
    Valeur cle;
    AB Pere,SAG, SAD;




};
```

le header : arborescences binaires

```
typedef int Valeur;
struct Sommet;
typedef Sommet* AB;
struct Sommet {
    Valeur cle;
    AB Pere,SAG, SAD;
    Sommet(Valeur);




};
```

le header : arborescences binaires

```
typedef int Valeur;
struct Sommet;
typedef Sommet* AB;
struct Sommet {
    Valeur cle;
    AB Pere,SAG, SAD;
    Sommet(Valeur);
    void GrefferSAG(AB);
    void GrefferSAD(AB);




};
```

le header : arborescences binaires

```
typedef int Valeur;
struct Sommet;
typedef Sommet* AB;
struct Sommet {
   Valeur cle;
   AB Pere,SAG, SAD;
   Sommet(Valeur);
   void GrefferSAG(AB);
   void GrefferSAD(AB);
   void SupprimerSAG();
   void SupprimerSAD();


};
```

le header : arborescences binaires

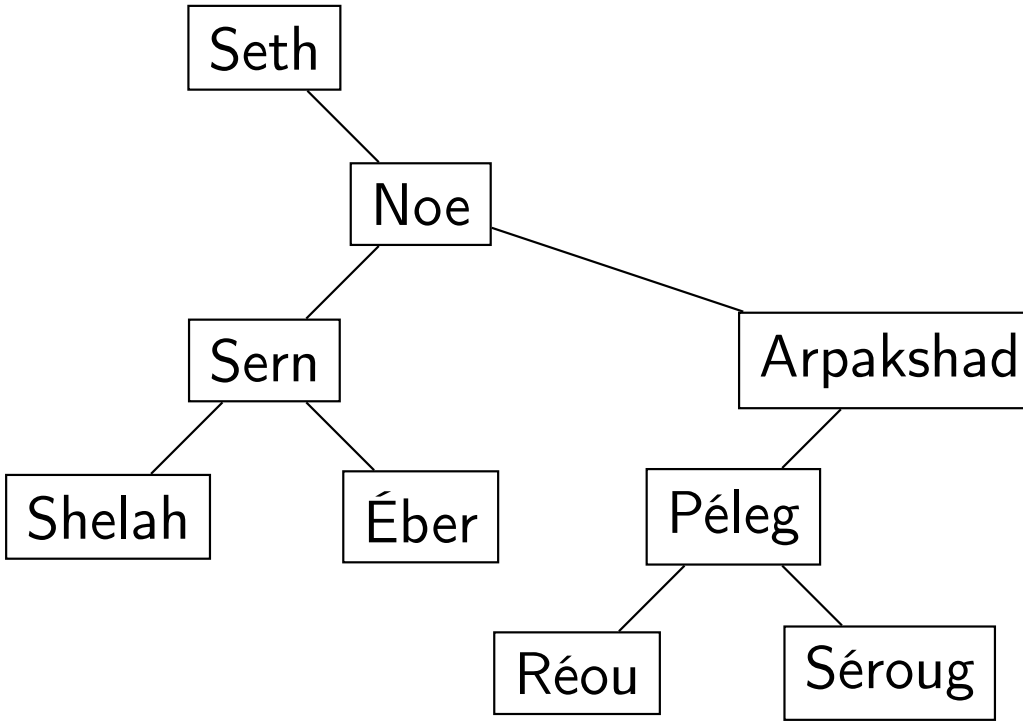
```
typedef int Valeur;
struct Sommet;
typedef Sommet* AB;
struct Sommet {
   Valeur cle;
   AB Pere,SAG, SAD;
   Sommet(Valeur);
   void GrefferSAG(AB);
   void GrefferSAD(AB);
   void SupprimerSAG();
   void SupprimerSAD();
   bool FeuilleP();

};
```

le header : arborescences binaires

```
typedef int Valeur;
struct Sommet;
typedef Sommet* AB;
struct Sommet {
   Valeur cle;
   AB Pere,SAG, SAD;
   Sommet(Valeur);
   void GrefferSAG(AB);
   void GrefferSAD(AB);
   void SupprimerSAG();
   void SupprimerSAD();
   bool FeuilleP();
   void RemplacerPourLePerePar(AB);
};
```

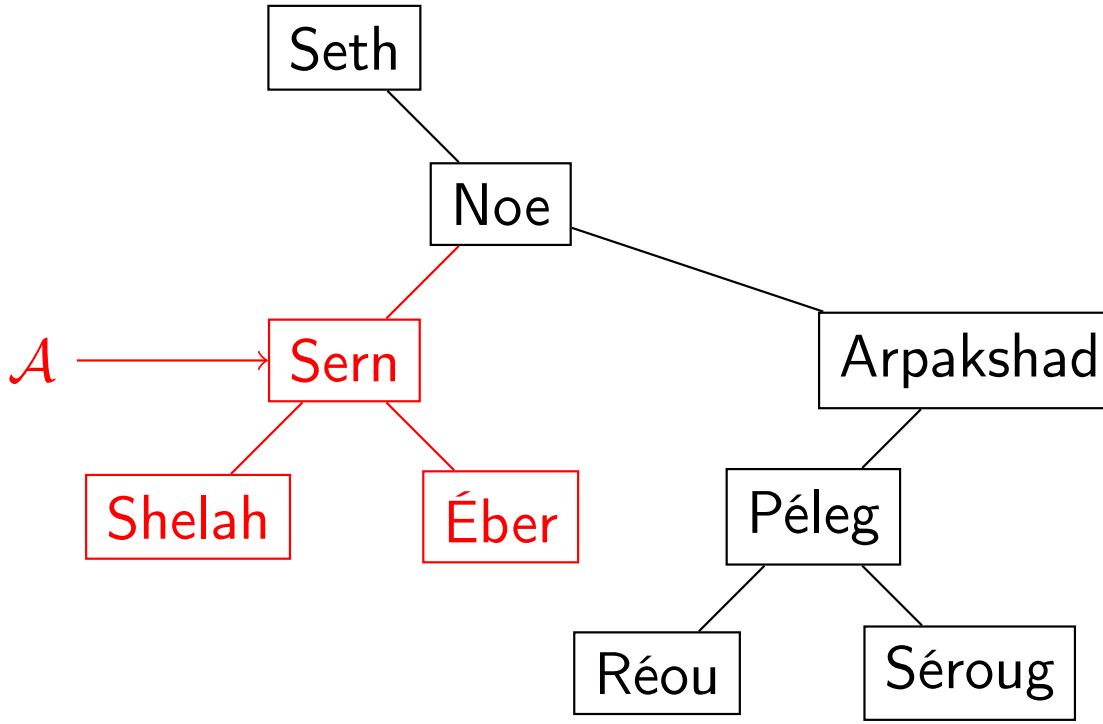
un exemple de l'utilisation de *RemplacerPourLePerePar*

Soit l'arborescence



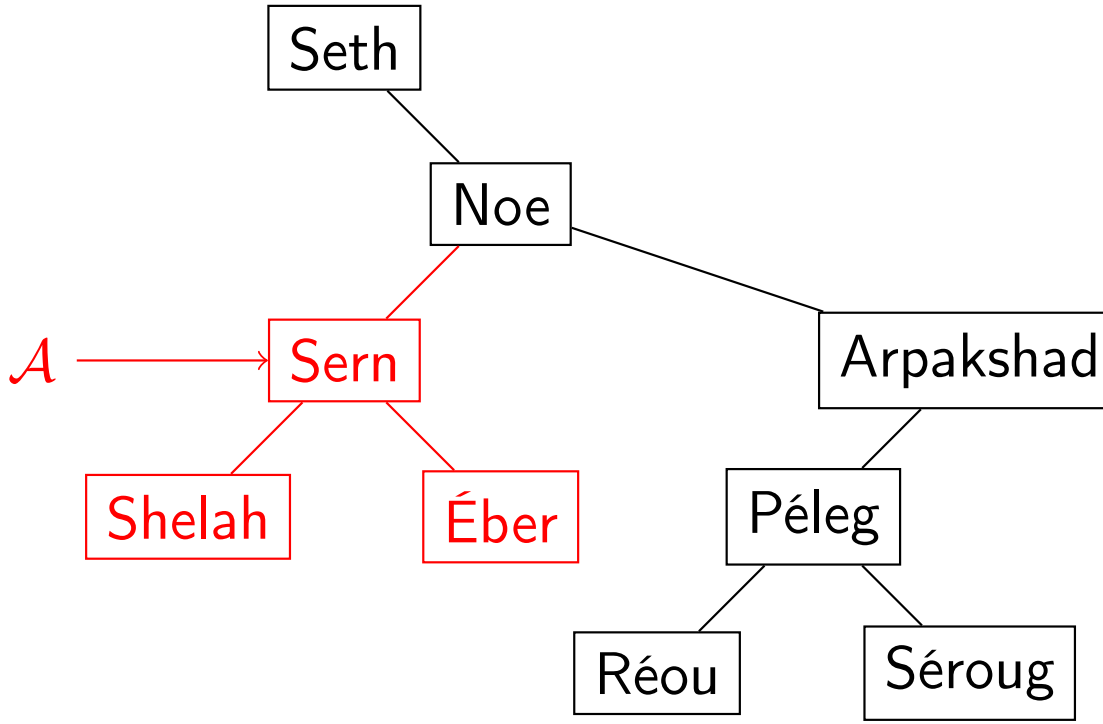
un exemple de l'utilisation de *RemplacerPourLePerePar*

Soit l'arborescence
et appelons la méthode *RemplacerPourLePerePar* sur la sous arborescence $\mathcal{A}$ de cle *Sern*



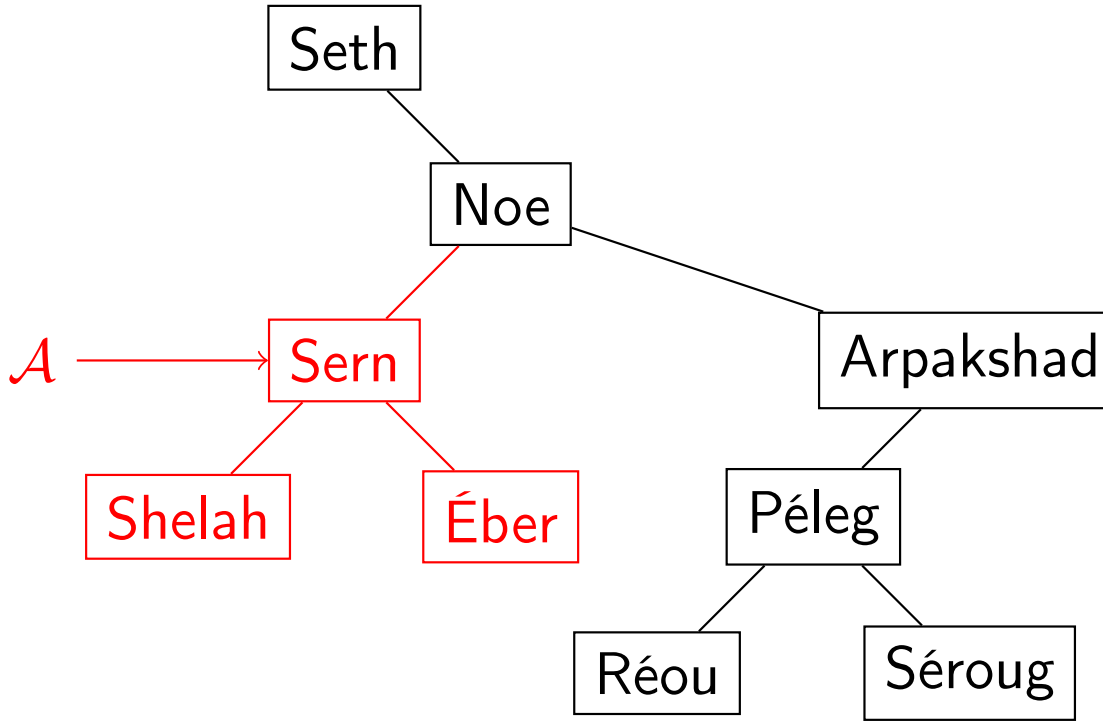
un exemple de l'utilisation de *RemplacerPourLePerePar*

Soit l'arborescence
et appelons la méthode *RemplacerPourLePerePar* sur la sous arborescence $\mathcal{A}$ de cle *Sern* : $\mathcal{A} \rightarrow$*RemplacerPourLePerePar*



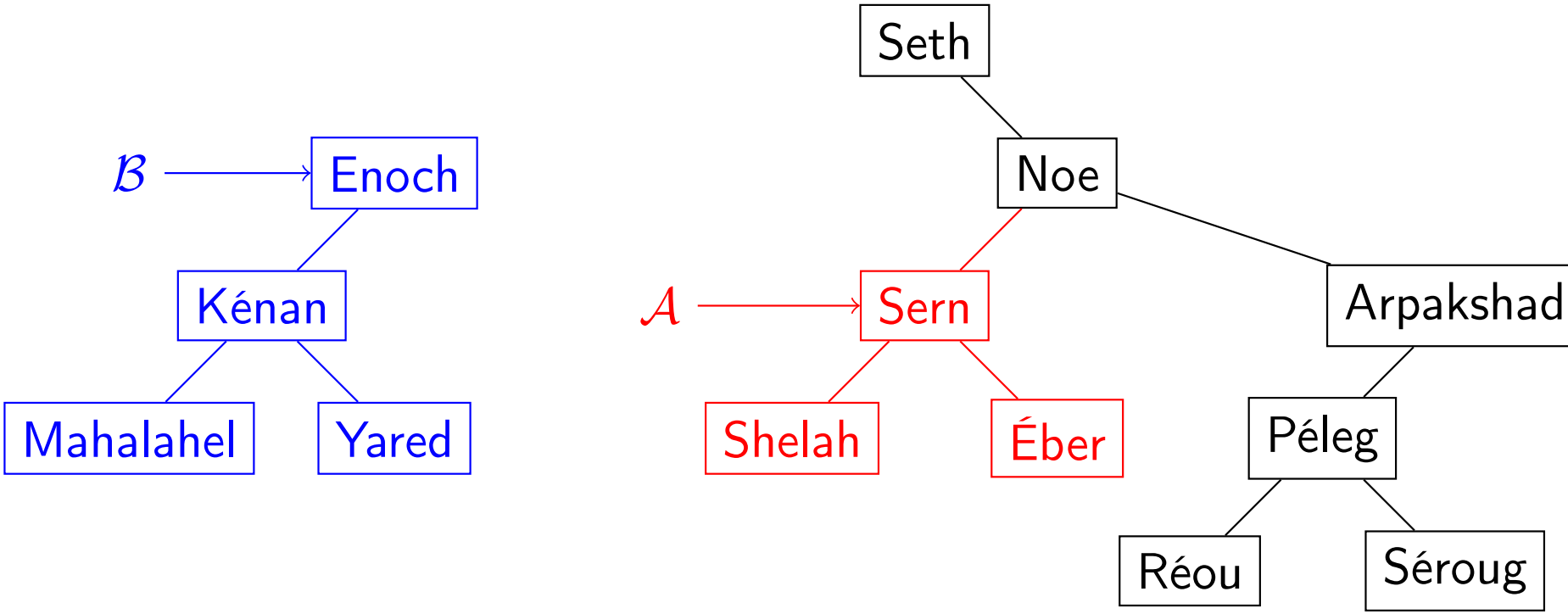
un exemple de l'utilisation de *RemplacerPourLePerePar*

Soit l'arborescence
et appelons la méthode *RemplacerPourLePerePar* sur la sous arborescence $\mathcal{A}$ de cle *Sern* : $\mathcal{A} \rightarrow$*RemplacerPourLePerePar* avec comme paramètre $\mathcal{B}$



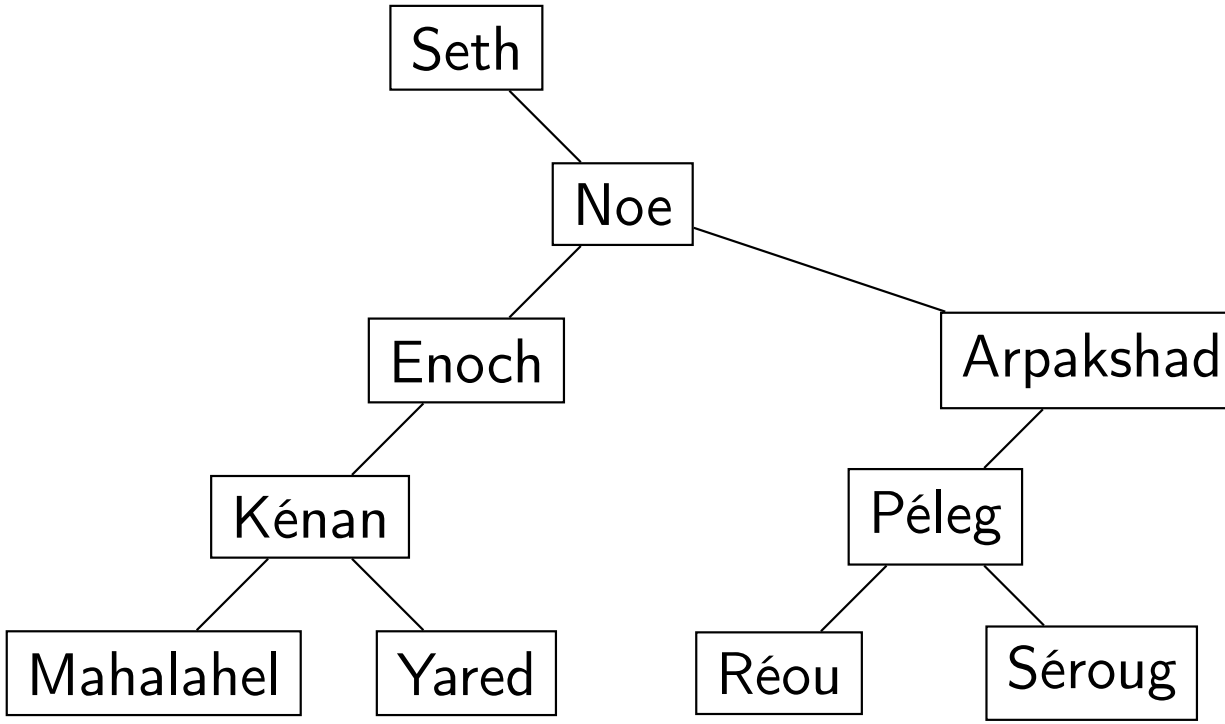
un exemple de l'utilisation de *RemplacerPourLePerePar*

Soit l'arborescence
et appelons la méthode *RemplacerPourLePerePar* sur la sous arborescence $\mathcal{A}$ de cle *Sern* : $\mathcal{A} \rightarrow$*RemplacerPourLePerePar*($\mathcal{B}$) avec comme paramètre $\mathcal{B}$



un exemple de l'utilisation de *RemplacerPourLePerePar*

l'appel $\mathcal{A} \rightarrow$ RemplacerPourLePerePar($\mathcal{B}$) transforme l'arborescence en



```
Sommet : :Sommet(Valeur v){

}

bool Sommet : :FeuilleP(){

}
```

```
Sommet : :Sommet(Valeur v){
   cle=v ;
}

bool Sommet : :FeuilleP(){

}
```

```
Sommet : :Sommet(Valeur v){
   cle=v ; SAG=NULL ; SAD=NULL ;
}

bool Sommet : :FeuilleP(){

}
```

```
Sommet : :Sommet(Valeur v){
   cle=v ; SAG=NULL ; SAD=NULL ; Pere=NULL ;
}

bool Sommet : :FeuilleP(){

}
```

```
Sommet : :Sommet(Valeur v){
   cle=v ; SAG=NULL ; SAD=NULL ; Pere=NULL ;
}

bool Sommet : :FeuilleP(){
  return (SAG==NULL) && (SAD==NULL) ;
}
```

```
void Sommet : :GrefferSAG(AB g){


}

void Sommet : :GrefferSAD(AB d){


}


void Sommet : :SupprimerSAG()   {

}
void Sommet : :SupprimerSAD()   {

}
```

```
void Sommet : :GrefferSAG(AB g){
    if (SAG) SAG→Pere=NULL ;

}

void Sommet : :GrefferSAD(AB d){



}


void Sommet : :SupprimerSAG()   {

}
void Sommet : :SupprimerSAD()   {

}
```

les algorithmes de manipulation de l'arborescence binaire

ajout et suppression d'une sous arborescence

```
void Sommet : :GrefferSAG(AB g){
    if (SAG) SAG→Pere=NULL ;
   SAG=g ; if (g) g→Pere=this ;
}

void Sommet : :GrefferSAD(AB d){



}


void Sommet : :SupprimerSAG()   {

}
void Sommet : :SupprimerSAD()   {

}
```

les algorithmes de manipulation de l'arborescence binaire

ajout et suppression d'une sous arborescence

```
void Sommet : :GrefferSAG(AB g){
    if (SAG) SAG→Pere=NULL ;
   SAG=g ; if (g) g→Pere=this ;
}

void Sommet : :GrefferSAD(AB d){
    if (SAD) SAD→Pere=NULL ;

}


void Sommet : :SupprimerSAG()   {

}
void Sommet : :SupprimerSAD()   {

}
```

les algorithmes de manipulation de l'arborescence binaire

ajout et suppression d'une sous arborescence

```
void Sommet : :GrefferSAG(AB g){
    if (SAG) SAG→Pere=NULL ;
   SAG=g ; if (g) g→Pere=this ;
}

void Sommet : :GrefferSAD(AB d){
    if (SAD) SAD→Pere=NULL ;
    SAD=d ; if (d) d→Pere=this ;
}



void Sommet : :SupprimerSAG()   {

}
void Sommet : :SupprimerSAD()   {

}
```

```
void Sommet : :GrefferSAG(AB g){
    if (SAG) SAG→Pere=NULL ;
   SAG=g ; if (g) g→Pere=this ;
}

void Sommet : :GrefferSAD(AB d){
    if (SAD) SAD→Pere=NULL ;
    SAD=d ; if (d) d→Pere=this ;
}



void Sommet : :SupprimerSAG()   {
   if (SAG) SAG→Pere=NULL ; SAG=NULL ;
}
void Sommet : :SupprimerSAD()   {
  if (SAD) SAD→Pere=NULL ; SAD=NULL ;
}
```

```
void Sommet : :RemplacerPourLePerePar(AB Ar){


}
```

```
void Sommet : :RemplacerPourLePerePar(AB Ar){
//le pere existe


}
```

```
void Sommet : :RemplacerPourLePerePar(AB Ar){
//le pere existe
   if (Ar) Ar→Pere=Pere ;

}
```

les algorithmes de manipulation de l'arborescence binaire
un peu plus compliqué

```
void Sommet : :RemplacerPourLePerePar(AB Ar){
//le pere existe
   if (Ar) Ar→Pere=Pere ;
   if (this==Pere→SAD)                    else
}
```

```
void Sommet : :RemplacerPourLePerePar(AB Ar){
//le pere existe
   if (Ar) Ar→Pere=Pere ;
   if (this==Pere→SAD)  Pere→SAD=Ar ; else
}
```

```
void Sommet : :RemplacerPourLePerePar(AB Ar){
//le pere existe
   if (Ar) Ar→Pere=Pere ;
   if (this==Pere→SAD)  Pere→SAD=Ar ; else  Pere→SAG=Ar ;
}
```