```
procedure Parcours(A) ;
  var X : sommet ;
  début
    [0]
    X := racine(A) ;
    [1]
    répéter
       si existe-fils(X) alors {
          [2]
          X := premier-fils(X) ;
          [3]
       }
       sinon {
          [4]
          tant que { X ≠ racine(A) et non existe-frère(X) } faire {
             [5]
             X := père(X) ;
             [6]
          }
          si existe-frère(X) alors {
             [7]
             X := frère(X) ;
             [8]
          }
       }
    jusqu'à X := racine(A) ;
    [9]
  fin ;
```

FIG. 6.3 – 6.2.2.1 Parcours en profondeur, itératif

```
procedure parcours(X) ;
begin
  [1]
  pour tout fils Y de X faire
    parcours(Y) ;
  [2]
end
```

```
void parcours(X) ;
{
  [1]
   l = liste-fils(X) ;
   if (len(l) == 0)
      [2]
      return ;
   for i in range(l) {
      parcours(element(l,i)) ;
      [3]
   }
   [4]
}
```

FIG. 6.4 – 6.2.2.2 Parcours en profondeur, récursif