procedure ***Parcours***$(\mathcal{A})$;
  var X : sommet ;
  début
    X := `racine`$(\mathcal{A})$;
    répéter
      si `existe-fils`(X) **alors** {
        empiler(X) ;
        X := `premier-fils`(X) ;
      }
      sinon {
        tant que $\left\{ \begin{array}{c} \mathrm{X} \neq \texttt{racine}(\mathcal{A}) \\ \textbf{et} \\ \textbf{non } \texttt{existe-frère}(\mathrm{X}) \end{array} \right\}$ **faire** {
          X := *dépiler*() ;
        }
        si `existe-frère`(X) **alors** {
          X := `frère`(X) ;
        }
      }
    jusqu'à X := `racine`$(\mathcal{A})$;
  fin ;

FIGURE 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]
}
```

FIGURE 6.4 – 6.2.2.2 Parcours en profondeur, récursif