Usage d'une pile dans les appels de procédures et fonctions

```
  algo test
  (* exemple pour montrer la pile des appels de procédures/fonctions *)

     fonction fact(n:entier):entier
     (* calcule la factorielle de n *)
     var n:entier
         f:entier
1    début
...      ...
n    fin

     procédure dialogue
     (* calcule la factorielle des nombres entrés au clavier *)
     (* jusqu'à ce qu'un nombre négatif soit entré           *)
     var n:entier
         f:entier
1    début
2        lire(n)
3        tant que n>0 faire
4             f ← fact(n)
5             écrire(f)
6             lire(n)
7        fin tant que
8    fin

  var f:entier
1 début
2     écrire("On commence...")
3     dialogue
4     f ← fact(12)
5     écrire(f)
6     écrire("On termine...")
7 fin
```

pile				test/4	test/4	test/4	test/4	
instruction	test/1	test/2	test/3	dialogue/1	dialogue/2	dialogue/3	dialogue/4	

pile	dialogue/5 test/4	dialogue/5 test/4	dialogue/5 test/4	test/4	test/4	test/4	test/4	
instruction	fact/1	fact/...	fact/n	dialogue/5	dialogue/6	dialogue/7	dialogue/3	

pile	test/4	... ...	test/4		test/5	test/5	test/5				
instruction	dialogue/...	...	dialogue/8	test/4	fact/1	fact/...	fact/n	test/5	test/6	test/7	

Pendant l'exécution de l'algorithme, on doit pouvoir déterminer à tout moment quel est la prochaine instruction à exécuter. En général, c'est celle qui suit dans le texte de l'algorithme. C'est un peu plus compliqué avec les constructions conditionnelles (`si alors fin si`) alternatives (`si alors sinon fin si`) et itératives (les boucles). Dans le cas de l'appel d'une procédure ou fonction, il faut "se rappeler" de reprendre l'exécution à l'instruction qui suit l'appel de la procédure ou fonction.

On peut numéroter les lignes d'instructions dans chaque algorithme/procédure/fonction et identifier les instructions par le nom de l'algorithme/procédure/fonction où se trouve l'instruction et le numéro de la ligne. Par exemple, dans l'algorithme ci-dessus, l'exécution commence à l'instruction `test/1`.

Les appels de procédures et fonctions peuvent être imbriqués : on appelle une fonction à l'intérieur d'une procédure par exemple. Pour chacun des appels, il faut se rappeler l'instruction à exécuter à la terminaison de cet appel. Par exemple, quand on appelle `fact` à la ligne `dialogue/4`, il faut noter que l'instruction à exécuter après la terminaison de l'appel de fonction est `dialogue/5`. Mais `dialogue` a elle même été appelée depuis `test/3`, à la terminaison de `dialogue` on devra reprendre l'exécution à `test/4`.

Pour cela on gère une pile des points de retour. À chaque appel, on empile le point de retour, à chaque retour, on dépile et reprend l'exécution à ce point.