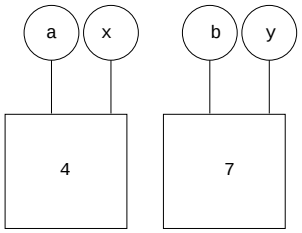
2.4.1 Pascal

Le langage Pascal comporte un mécanisme primitif appelé le passage d'arguments par référence, *ou* par variable. *Ainsi, dans la définition de la procédure* `swap`, *on peut faire précéder chaque argument du mot clé* `var`.

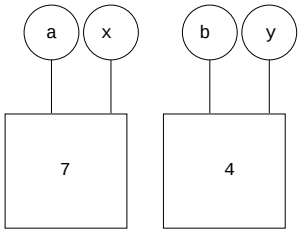
```
procedure swap (var x : integer, var y : integer) ...
```

Quand un argument d'une fonction est ainsi déclaré en passage par référence, on ne peut appliquer cette fonction qu'à une variable. Ainsi, on peut écrire `swap(a,b)` *mais pas* `swap(4,7)`, *ni* `swap(2 * a,b)`.

Quand on appelle la procédure `swap(a,b)`, *au lieu d'associer les variables* `x` *et* `y` *à de nouvelles références et d'associer dans la mémoire ces références aux valeurs des arguments réels de la procédure,* `4` *et* `7`, *on associe les variables* `x` *et* `y` *aux références associées aux variables données en argument à la procédure. Ainsi, quand on appelle la procédure* `swap(a,b)` *dans un environnement* `e = [a = r1, b = r2]` *et une mémoire* `m = [r1 = 4, r2 = 7]`, *au lieu de fabriquer l'environnement* `[a = r1, b = r2, x = r3, y = r4]` *et la mémoire* `[r1 = 4, r2 = 7, r3 = 4, r4 = 7]`, *on fabrique l'environnement* `[a = r1, b = r2, x = r1, y = r2]` *et on garde la mémoire* `[r1 = 4, r2 = 7]`.



De ce fait, la procédure `swap` *intervertit le contenu des références* `r1` *et* `r2` *et non celui des références* `r3` *et* `r4`



et après l'exécution de la procédure, les contenus des références associées aux variables `a` *et* `b` *ont bien été intervertis.*

Pouvoir expliquer ce mécanisme du passage par référence est la principale motivation pour décomposer l'état en un environnement et une mémoire en