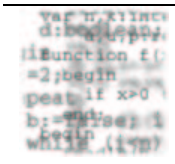

TurboPascal - Prépa HEC Ipecom

Fonctions et procédures 2/2

notes de cours - vendredi 05 avril

1 Imbrication

On a vu qu'on peut définir dans un programme des variables, des procédures, des fonctions, et qu'on peu également définir dans ces procédures et fonction des variables, procédures et fonctions locales, et ainsi de suite. Chacun de ces objets est local pour la structure immédiatement supérieure, c'est-à-dire la structure dans laquelle il et défini; il ne peut donc être appelé ou manipulé que par des structures définies au même niveau, ou au niveau immédiatement supérieur. Ainsi, si `P` est le programme, `A`, `B`, `C` des procédures et `p1`, `p2`, `a1`, `a2`, `b1`, `b2`, `c1`, `c2`, des variables définies dans ces structures, `C` étant défini dans `C`, il est important de bien voir la structure, où les `[]` délimitent la portée des objets:

```
P[p1,p2,A[a1,a2],B[b1,b2,C[c1,c2]]],
```

2 Appels de procédure et fonction

- Lors d'un passage par adresse, il est nécessaire que le paramètre de l'appel de la fonction ou procédure soit une variable [`a`], et pas une valeur [`2`], pour que le paramètre puisse être modifié par la fonction ou la procédure.
- Lors d'un passage par valeur, il est possible que le paramètre soit une valeur, même une expression [`x+3*z*(ln(a)-b)`].

3 Récursivité

On parle de récursivité dans une procédure [de même que dans une fonction] lorsque le corps de la procédure comprnd un appel à cette même procédure. Cela se prète bien à certains types de tâches, proches de la récurrence mathématique, comme la factorielle, ou des suites définies par récurrence. Deux remarques sont nécessaires:

- La procédure s'appellant elle-même, on a une boucle, qui doit se terminer. Il faut donc toujours prévoir un test de fin au bouclage [nombre d'itérations atteint, ...].
- Généralement, un algorithme récursif est plus économe en opérations et affectations, donc plus rapide, mais BEAUCOUP plus gourmand en mémoire. En effet, si une procédure `A`, avec ses variables locales `v1` et `v2` s'appelle 100 fois, il y aura 100 procédures `A` en exécution simultanée jusqu'à ce que la boucle se termine, et donc les variables locales pour chacun des 100 appels simultanés de `A`:

```
A[v1,v2,A[v1,v2,A[v1,v2,A[v1,v2,A[v1,v2,A[v1,v2,A[v1,v2,...
...A[v1,v2,A[v1,v2,A[v1,v2,A[v1,v2,A[v1,v2,A[...]]]]]]]]]]]]
```