D Passage de tableaux en arguments

En MPD, il est possible pour une procédure (ou fonction) de recevoir en argument (paramètre formel) un tableau de taille arbitraire, c'est-à-dire, pour lequel les bornes du tableau ne sont pas indiquées explicitement dans la déclaration du paramètre formel. Ceci se fait en utilisant «*» comme bornes du tableau.

Par exemple, soit la procédure suivante, où `a` est un tableau passé par valeur (mode `val`, qui est le mode par défaut) :

```
procedure imprimerBornes( string[10] nom, int a[*] )
{
  printf( "%s[%d:%d]\n", nom, lb(a), ub(a) );
}
```

Dans ce cas, les bornes du tableau à l'intérieur de la procédure vont toujours de `1` à n, où n représente le nombre d'éléments du tableau, et ce peu importe la façon dont le tableau passé en argument est déclaré. Par exemple, les appels suivants produiraient les résultats indiqués en commentaire :

```
int a1[10]    = ([10] 0);
int a2[0:99]  = ([100] 0);
int a3[23:33] = ([11] low(int));

imprimerBornes( "a1", a1 );  # => a1[1:10]
imprimerBornes( "a2", a2 );  # => a2[1:100]
imprimerBornes( "a3", a3 );  # => a3[1:11]
```

Le comportement est exactement le même si le mode de passage du tableau est `var` (variable) ou `res` (résultat). Par contre, si le mode de passage du paramètre est `ref`, alors les bornes sont plutôt *celles du tableau passé en argument* :

```
procedure imprimerBornes( string[10] nom, ref int a[*] )
{
  printf( "%s[%d:%d]\n", nom, lb(a), ub(a) );
}

int a1[10]    = ([10] 0);
int a2[0:99]  = ([100] 0);
int a3[23:33] = ([11] low(int));

imprimerBornes( "a1", a1 );  # => a1[1:10]
imprimerBornes( "a2", a2 );  # => a2[0:99]
imprimerBornes( "a3", a3 );  # => a3[23:33]
```

La même règle s'applique pour les *tranches de tableaux* — par exemple, une expression de la forme «`a1[3:6]`», qui dénote un sous-tableau (série d'éléments adjacents) de `a1` — à la différence qu'une telle tranche *ne peut jamais être passée par référence.*