I. Les tableaux

A. Définition Usuelle : Array of

1. Vecteur

a) Taille fixe et connue lors de la compilation

Lorsque la taille d'un tableau est connue à l'avance, la déclaration et l'utilisation du tableau est simple :

Déclaration :

```
Var T : Array[0..5] of Double ;
```

T est ici un tableau de 6 variables double. Le premier élément est T[0] et le dernier est T[5].

Utilisation :

```
T[0] := 2.34534 ;
T[1] := ln(2) ;
T[2] := T[0] / T[1] * exp(T[0] + 1) ;
…
for i :=0 to 5 do T[i] := i ;
```

ici il y a conversion implicite de i integer en double.

Libération de la mémoire : Automatique (Out of Scope)

b) Taille inconnue lors de la compilation

Lorsque la taille d'un tableau est connue à l'avance, la déclaration et l'utilisation du tableau sont différentes :

Déclaration :

```
Var T : Array of Double ;
```

T est ici un tableau de variables double de taille non fixée.

Initialisation Mémoire : Il faut d'abord créer la place mémoire qu'occupera le tableau

```
SetLength(T,6) ;
```

Ici on a défini un tableau de 6 double. Le premier élément est T[0] et le dernier est T[5].

L'utilisation est ensuite identique :

```
T[0] := 2.34534 ;
T[1] := ln(2) ;
T[2] := T[0] / T[1] * exp(T[0] + 1) ;
…
for i :=0 to 5 do T[i] := i ;
```

ici il y a conversion implicite de i integer en double.

Libération de la mémoire : Non Automatique. Il faut libérer la mémoire à la fin de l'utilisation du tableau. L'instruction est très simple :

```
T := nil ;
```

2. Matrice

a) Taille fixe et connue lors de la compilation

Lorsque la taille d'un tableau est connue à l'avance, la déclaration et l'utilisation du tableau est simple :

Déclaration :

```
Var T : Array[0..5, 0..3] of Double ;
```

T est ici un tableau de 6x4 variables double. Le premier élément est T[0,0] et le dernier est T[5,3].

Utilisation :

```
T[0,0] := 2.34534 ;
T[1,2] := ln(2) ;
T[2,0] := T[0,0] / T[1,1] * exp(T[0,3] + 1) ;
…
for i :=0 to 5 do T[i,0] := i ;
```

ici il y a conversion implicite de i integer en double.

Libération de la mémoire : Automatique (Out of Scope)

b) Taille inconnue lors de la compilation

Lorsque la taille d'un tableau est connue à l'avance, la déclaration et l'utilisation du tableau sont différentes :

Déclaration :

```
Var T : Array of Array of Double ;
```

T est ici une matrice de variables double de taille non fixée.

Initialisation Mémoire : Il faut d'abord créer la place mémoire qu'occupera le tableau

```
SetLength(T,6,4) ;
```

Ici on a défini un tableau de 6x4 double. Le premier élément est T[0,0] et le dernier est T[5,3].

L'utilisation est ensuite identique :

```
T[0,0] := 2.34534 ;
T[1,2] := ln(2) ;
T[2,0] := T[0,0] / T[1,1] * exp(T[0,3] + 1) ;
…
for i :=0 to 5 do T[i,0] := i ;
```

ici il y a conversion implicite de i integer en double.

Libération de la mémoire : Non Automatique. Il faut libérer la mémoire à la fin de l'utilisation du tableau. L'instruction est très simple :

```
T := nil ;
```

B. Définition par les pointeurs

1. Vecteur

Seul le cas à taille variable est intéressant.
Tout d'abord, il faut déclarer 2 types :

```
Const NMAX = 1000000 ;
```

cette constante sera la borne supérieure maximale du tableau

```
Type
```

```
        TabDouble = Array[0..NMAX] of Double ;
        PtrDouble = ^TabDouble;
```

Puis on déclare les deux types précédents. Le type PtrDouble est un pointeur sur un tableau de taille variable que nous définirons nous même lors de l'exécution du code.

Utilisation :

```
Var     Ptab : PtrDouble ;
        NbValue : Integer ; le nombre de valeurs dans le tableau
Begin
        …
        NbValue := 10 ;
        GetMem(Ptab, NbValue * SizeOf(Double)) ; initialisation de la mémoire
        …
        Ptab^[0] := 3;
        …
        Ptab^[NbValue-1] := ln(3);
        …
        FreeMem(Ptab, NbValue * SizeOf(Double)) ; libération de la mémoire
End ;
```

Remarquez le "^" entre le nom de la variable et les crochets. Cela sert à déréférencer le pointeur. Ptab[0] correspond à une adresse mémoire, Ptab^[0] se situe à l'adresse mémoire Ptab[0] et contient dans notre exemple le nombre 3 .

2. Matrice

La définition est relativement simple : un pointeur sur un tableau de pointeurs.

```
Const NMAX = 1000000 ; cette constante sera la borne supérieure maximale du tableau
Type
        TabDouble = Array[0..NMAX] of Double ;
        PtrDouble = ^TabDouble;
        TabMatrice = Array[0..NMAX] of PtrDouble;
        PtrMatrice = ^TabMatrice;
```

Utilisation :

```
Var     PMat : PtrMatrice ;
        NbLignes, NbColonnes : Integer ; le nombre de valeurs dans le tableau
Begin
        …
        NbValue := 10 ;
-----------------------------------------
initialisation de la mémoire
-----------------------------------------
        GetMem(Ptab, NbLignes * SizeOf(PtrDouble)) ; création du vecteur de PtrDouble
        For i :=0 to NbLignes-1 do création de la mémoire de chaque vecteur de PtrDouble
                GetMem(Ptab, NbColonnes  * SizeOf(Double)) ;
-----------------------------------------
        …
```

```
        Ptab^[0]^[0] := 3;
        …
        Ptab^[ NbLignes –1]^[NbColonnes-1] := ln(3);
        …
-----------------------------------------
```
libération de la mémoire : L’ordre est inversé !!!
```
-----------------------------------------
        For i :=0 to NbLignes-1 do
                FreeMem (Ptab, NbColonnes  * SizeOf(Double)) ;
        FreeMem (Ptab, NbLignes * SizeOf(PtrDouble)) ;
-----------------------------------------
End ;
```