Feuille de TD/TP n° 7

Algorithmes et tableaux

Pour chacun des exercices de cette feuille, commencer par écrire les algorithmes sur papier (en langage algorithmique simple) et déterminer leur complexité avant de passer au code Ada.

Exercice 1 : Définir un sous-type `TIndex` des entiers naturels, et un sous-type `TValue` des entiers relatifs. Définir un type tableau non contraint `TArray` indicé par `TIndex` et contenant des éléments de type `TValue`. Définir les fonctions et procédures suivantes :

- `function Equal (T1, T2 : TArray) return Boolean` : renvoie vrai si les tableaux sont égaux (mêmes valeurs dans l'ordre), sans utiliser le test d'égalité primitif entre tableaux. Attention ! les plages d'indices peuvent ne pas coïncider.
- `function Get_Nth (T : TArray, N : Natural) return TValue` : renvoie le `N`-ième élément du tableau `T` (en commençant à 1 pour la première case). Astuce : utiliser judicieusement `TIndex'Succ`.
- `function Equal_Bis (T1, T2 : TArray) return Boolean` : renvoie vrai si les tableaux sont égaux (similaire à `Equal`). Utiliser la fonction précédente.
- `function Sorted (T : TArray) return Boolean` : renvoie vrai si le tableau est trié dans l'ordre croissant. Proposer une version itérative et une version récursive.
- `function Max (T : TArray) return TValue` : renvoie le plus grand élément du tableau. Proposer une version itérative et une version récursive.
- `function Max_2 (T : TArray) return TArray` : renvoie les deux plus grands éléments du tableau (le plus grand dans la première case). Proposer une version itérative et une version récursive.
- `function Max_N (T : TArray, N : Natural) return TArray` : renvoie les `N` plus grands éléments du tableau (le plus grand dans la première case, puis en ordre décroissant). Proposer une version itérative et une version récursive.

  Qu'obtient-on si l'on appelle cette fonction avec pour `N` la taille du tableau ?

Exercice 2 : En se souvenant que le type `String` est un type tableau non contraint de caractères indicés par des entiers, définir une fonction `Palindrome` prenant en paramètre une chaîne de caractère et renvoyant vrai si et seulement si celle-ci est un palindrome (comme les mots "kayak" ou "ressasser" par exemple).

Exercice 3 : le triangle de Pascal est un tableau donnant les coefficients binomiaux sous forme d'un triangle où chaque ligne contient le développement du binôme à un exposant donné. Voici les 5 premières lignes du triangle :

```
1     1
1     2     1
1     3     3     1
1     4     6     4     1
1     5    10    10     5     1
```

Le triangle est construit très simplement d'une ligne à l'autre de la façon suivante. Dans la ligne suivante, on met la valeur 1 dans la première et la dernière case, puis, pour chacune des autres cases, on met la somme des deux valeurs : de la case de même colonne et de la case de colonne précédente, dans la ligne précédente.

Afin de réaliser l'affichage du triangle, définir une fonction `Next_Line` qui prend en paramètre une ligne et renvoie en résultat la ligne suivante. Puis utiliser cette fonction pour définir une procédure récursive `Pascal` qui prend en paramètre une ligne et en entier : elle affiche la ligne reçue en paramètre, puis si l'entier est strictement positif, elle se rappelle elle-même avec le résultat de la fonction `Next_Line` et l'entier décrémenté de 1.

Alternativement, proposer un algorithme qui travaille directement sur une matrice.