# Travaux Pratiques d'algorithmique n°6

**Cours d'Informatique de Deuxième Année**

*—DEUG Sciences S3-MI—*

---

## Tableaux de structures

Dans cette séance, nous allons manipuler des "fiches" concernant des personnes et représentées par des structures. Après avoir implémenté les fonctions de base (saisie, affichage, comparaison, copie), nous utiliserons ces fonctions pour gérer un ensemble de personnes. Nous trierons les éléments de cet ensemble selon deux méthodes. Enfin, nous étudierons les entrées/sorties des données sur disque.

---

Le type `Personne` utilisé durant ce TP est défini de la façon suivante :

```
struct personne
{
  char         Nom[MAXSTR];
  char         Prenom[MAXSTR];
  unsigned int Age;
};

typedef struct personne Personne;
```

où `MAXSTR` est la taille maximale d'une chaîne de caractères.

▸ **Exercice 1.** *Écrire les fonctions :*

- `SaisiePersonne` *qui met à jour une fiche passée en argument avec les valeurs saisies par l'utilisateur ;*
- `AffichePersonne` *qui affiche le contenu d'une fiche passée en argument ;*
- `ComparePersonnes` *qui compare les deux fiches passées en argument selon le nom, le prénom puis l'âge. Elle renverra un nombre négatif, nul ou positif selon que la première fiche est "inférieure", "égale" ou "supérieure" à la seconde ;*
- `CopiePersonne` *qui copie dans la première fiche la seconde fiche passée en argument.*

*Comment créer un tableau de* `MAXENS` *personnes (où* `MAXENS` *est une constante) ?*
*Définir un type* `Ensemble` *contenant un tel tableau et la taille qui lui est associée. En déduire les fonctions* `void InitEnsemble(Ensemble *e)`, `int AjoutePersonne(Personne p,Ensemble *e)` *et* `void AfficheEnsemble(Ensemble e)`.

▸ **Exercice 2.** ***(Construction d'un ensemble trié)*** *Nous allons construire un tableau trié de $n$ éléments en considérant que si l'on dispose d'un tableau trié de $n-1$ éléments, il suffit d'insérer le $n^{ieme}$ élément à la bonne place dans ce tableau. Ainsi, le tableau trié de $n$ éléments sera construit par insertions successives des $n$ éléments.*
*En utilisant les fonctions précédentes, écrire une fonction* `int InserePersonne(Personne f, Ensemble *e)` *qui insère la fiche* `f` *à sa place (selon l'ordre croissant) dans l'ensemble* `e`*. Cette fonction renvoie 1 ou 0 selon que l'insertion à été effectuée ou pas. Évaluer la complexité de l'insertion et celle de $n$ insertions successives.*

▸ **Exercice 3.** ***(Tri à bulles)*** *Le tri à bulles consiste à parcourir le tableau en permutant deux à deux les éléments qui ne sont pas dans l'ordre souhaité. Ainsi, lors du premier passage on fera remonter (comme une bulle) l'élément le plus grand en haut du tableau. Le second parcours se fait donc sur un tableau de taille $n-1$ et ainsi de suite...*
*Écrire la fonction* `void TriBulles(Ensemble *e)` *qui trie l'ensemble* `e`*.*
*Évaluer sa complexité.*

▸ **Exercice 4.** *Dans cet exercice, nous allons utiliser les fonctions système (*`fopen`*,* `fclose`*,* `fread` *et* `fwrite` *permettant d'écrire et lire des données dans un fichier.*
*La fonction* `fopen` *permet d'ouvrir un fichier selon différents modes. Les principaux modes utilisés sont : "r" pour la lecture, "w" pour l'initialisation/écriture et "a" pour la mise à jour.* `fopen` *renvoie un pointeur sur une structure de type* `FILE`*, appellée flux (stream en anglais).*

- *Écrire une fonction* `unsigned int EcritEnsemble(Ensemble *e, char *fic)` *qui écrit les fiches de l'ensemble* `e` *dans le fichier ayant pour nom* `fic` *;*
- *écrire une fonction* `unsigned int LitEnsemble(Ensemble *e, char *fic)` *qui lit des fiches dans le fichier* `fic` *(quel que soit leur nombre) et les place dans un tableau* `tab` *(de taille maxi* `MAXENS`*) ;*
- *que se passe-t-il si l'on utilise plusieurs fois la fonction* `EcritEnsemble` *sur le même fichier ? Comment modifier le programme pour résoudre ce problème ?*

*Les fonctions devront renvoyer le nombre de fiches lues ou écrites, et 0 si elles ne se sont pas déroulées correctement.*