

# L3 CDA – TP 6 C (4h)

**points :** classes mémoire, compilation séparée, structures.

## 1 Classes mémoire

**Q1:** Proposer une fonction qui propose, à chaque appel, un nombre aléatoire différent, compris entre 1 et 20. Au bout de 20 appels, elle retourne -1. Un tirage aléatoire peut être fait avec `random()`. L'implémentation sera faite 2 fois, la première en utilisant des variables globales, et la seconde sans variables globales.

**Q2:** Testez `strtok()` (regardez la page de manuel correspondante). Comprenez comment la fonction est implantée.

## 2 Type Abstrait de Données *Ensemble*

**Q3:** Implanter un TAD ensemble d'entiers positifs, avec allocation statique de la taille maximale de l'ensemble. La spécification des fonctions est donnée dans le fichier d'entête ci-dessous :

```
// ens.h

// le nombre d'elements alloues est fixe (version statique)
#define MAXENS 128
typedef struct {
    int nb;           // nombre d'elements actuellement contenus dans l'ensemble
    int donnees[MAXENS]; // les donnees elle memes
} ens_t;

void ensNouv(ens_t *e);           // initialise un ensemble
int ensIns(ens_t *e, int val);    // insere val dans l'ensemble, retourne 1 si succes, 0 si echec
int ensTrouve(ens_t *e, int val); // retourne l'index de la premiere occurence de val dans
                                // e (-1 si absent)
int ensSuppr(ens_t *e, int val);  // supprime la premiere occurence de val dans e (renvoie
                                // son index, et -1 si absent)
int ensSupprTous(ens_t *e, int val); // supprime toutes les occurences de val dans e (renvoie
                                // le nombre d'occurences supprimees)
void ensReset(ens_t *e);          // vide e. La reimplenter sous forme de macro.
void ensAff(ens_t *e);            // affiche le contenu d'un ensemble sur la sortie standard

// quelques fonctionalites supplementaires
// implantees sous forme de macros
#define ensContient(e, val) (ensTrouve(e, val) != -1 ? 1 : 0) // vrai si val present dans e
#define ensNb(e) (e->nb) // taille de e
#define ensVide(e) (ensNb(e) == 0 ? 1 : 0) // vrai si e vide
#define ensInsUnique(e, val) (ensContient(e, val) ? 0 : (ensIns(e, val), 1)) /* insere val dans e sans
                                doublon. renvoie vrai
                                si insertion faite
                                faux si val presente */
```