Travaux Pratiques d'algorithmique n°5

Cours d'Informatique de Deuxième Année

—DEUG Sciences M3-MI—

Les structures

Le but de ce TP est d'effectuer des opérations sur des **ensembles**. Un ensemble sera représenté par une structure à deux champs :

- la taille de l'ensemble ;
- le contenu de l'ensemble, à savoir un tableau d'entiers non trié.

On étudiera deux méthodes :

- dans un premier temps, on utilisera des tableaux de taille fixe ;
- dans un second temps, on préférera manipuler des tableaux de taille variable (on fera alors appel à une allocation dynamique).

▸ **Exercice 1.** *La structure suivante permet de définir un ensemble comme un tableau de* `MAX` *entiers ainsi que le nombre d'éléments "utilisés" du tableau.*

```
typedef struct tableau
{
  int Tab[MAX];
  int Nb_el;
} ENS;
```

A partir de cette structure, programmer les fonctions suivantes :

- `int Est_Vide(ENS e)` *qui indique si le tableau contenu dans la structure, est vide, (retourne 1 si vrai, 0 sinon) ;*
- `int Recherche(ENS e,int val)` *qui teste si l'élément* `val` *appartient à l'ensemble* `E`*, (retourne 1 si vrai, 0 sinon) ;*
- `int Insere_Valeur(ENS *e,int val)` *qui insère un élément de valeur* `val` *dans le tableau de la structure ;*
- `void Affiche_Tableau(ENS e)` *;*
- `int Enleve_Elem(ENS *e,int ieme)` *qui supprime le k-ième élément du tableau.*

▸ **Exercice 2.** *On considère maintenant que* ***les tableaux sont triés****. La structure reste inchangée, mais certaines fonctions requièrent des modifications. Réécrire ces fonctions.*

On choisit maintenant la structure de données suivante pour les ensembles :

```
typedef struct {
        unsigned int NbElts;
        int * Tab;
        } Ensemble;
```

Les tableaux ne sont pas triés.

▸ **Exercice 3. (allocation)** *Ecrire les fonctions* `int AllouerEns(Ensemble *E,unsigned int nb)`, `int LibererEns(Ensemble *E)`, `int ReallouerEns(Ensemble *E, unsigned int nb)` *et* `int RecopierEns(Ensemble E, Ensemble *Recop)` *pour respectivement allouer de la place pour le tableau d'un ensemble, libérer cet espace alloué, réallouer pour le même ensemble un tableau de taille différente et faire une recopie d'un ensemble dans un autre ensemble.*

▸ **Exercice 4.** *Ecrire les fonctions suivantes :*

- `int MembreEns(int elt, Ensemble E)` *qui teste si l'élément* `elt` *appartient à l'ensemble* `E` *;*
- `void SaisirEns(Ensemble *E)` *qui saisit un ensemble d'entiers ;*
- `void AfficherEns(Ensemble E)` *qui affiche le contenu de l'ensemble* `E`.

▸ **Exercice 5. (union, intersection, difference)** *Ecrire les fonctions* `Union`, `Intersection`, `Difference` *et* `DifferenceBis` *qui produisent l'union, l'intersection et la différence de deux ensembles.*
Indication :

$$Difference(E1, E2) = E1 - E2 \ ,$$

$$DifferenceBis(E1, E2) = Union(E1 - E2, E2 - E1) \ .$$

Evaluer la complexité des algorithmes.

▸ **Exercice 6. (inclusion, égalité, complémentaire)** *Ecrire les fonctions suivantes :*

- `int EnsInclus(Ensemble E1,Ensemble E2)` *qui teste si* `E1` *est inclus dans* `E2` *;*
- `int EnsEgaux(Ensemble E1,Ensemble E2)` *qui teste si les deux ensembles sont égaux ;*
- `int Complementaire(Ensemble E1,Ensemble E2,Ensemble *Result)` *qui retourne le complémentaire de* `E1` *par rapport à* `E2`.

Evaluer la complexité des algorithmes.