

## Allocation dynamique et liste chaînée

Le but de ce TP est la manipulation des pointeurs sur des types structurées allouées dynamiquement (malloc et free). La liste chaînée est la structure de donnée employée pour appliquer ces concepts.

Merci de vous reporter au sujet du premier TP pour le protocole de rendu des exercices. Dans les exercices, on utilisera les types de données suivants :

```
typedef struct
{
    char nom[30];
    int anneeNaissance;
    char telephone[20];
} personne;

typedef struct cellule
{
    personne val;
    struct cellule* prochaineCellule;
} cellule;
```

### Exercices

1. Ecrire une procédure `int saisirPersonne( personne* p )` qui demande à l'utilisateur de saisir les informations d'une personne. Ces informations seront stockées dans l'objet de type `personne` pointé par `p`. Cette procédure retourne 0 si l'utilisateur entre un symbole spécial pour signifier qu'il n'a plus de données à saisir, 1 sinon.
2. Ecrire une procédure `void afficherPersonne( personne p )` qui affiche les informations de la personne `p`.
3. Ecrire une procédure `int estIdentique( personne p1, personne p2 )` retournant 0 si les deux objets `p1` et `p2` ont les mêmes valeurs.
4. Ecrire une procédure `cellule* creerCellule( personne p )` qui retourne un pointeur vers une cellule dont le champ `val` est instancié avec les valeurs de `p`. Indication : `man malloc`.
5. Ecrire une procédure `cellule* saisirListeCellule( void )` qui demande à l'utilisateur de taper une liste de personnes et qui remplit une liste de cellules en utilisant le champ `prochaineCellule`. Le premier élément de la liste est renvoyé par `saisirListeCellule`.
6. Ecrire une procédure `void afficherListeCellule( cellule* liste )` qui affiche les personnes contenues dans `liste`, où `liste` est un pointeur vers le premier élément de la liste chaînée de cellules.
7. Ecrire une procédure `void separation( int anneeNaissance, cellule* liste )` qui supprime de la liste chaînée de cellules `liste` les éléments `cellule` dont la personne est née avant `anneeNaissance`. Les éléments supprimés devront être libérés de la mémoire. Indication : `man free`.
8. Traduire le texte complet de ce TP en code C++ avec des classes (au lieu des définitions `typedef struct`) et l'utilisation des opérateurs `new/delete` (au lieu de `malloc/free`). Pour cela, vous créez un nouveau projet Eclipse configuré pour le C++! Les définitions des structures de données seront désormais :

```
class personne
{
public:
    char nom[30];
    int anneeNaissance;
    char telephone[20];
    void afficherPersonne(void);
    bool estIdentique( personne p2 );
};

class cellule
{
public:
    personne val;
    cellule* prochaineCellule;
    void afficherListeCellule(void);
};
```

Remarque : les procédures des exercices 3 et 4 seront des méthodes de la classe `personne`. La procédure de l'exercice 6 sera une méthode de la classe `cellule`. Commentez les deux versions : laquelle vous semble la plus lisible? Pourquoi?