

Feuille d'exercices n°7

Les listes chaînées en C

Exercice 1:

Soit la structure suivante

```
-----  
struct Address {  
    int no;  
    char *rue;  
    int codepostale;  
    char *ville;  
    char *pays;  
};  
-----
```

- Écrire une fonction `Address createAddress()` pour la création d'une adresse.
- Écrire une fonction `affichageAddress(Address *ad)` qui affiche le contenu d'une adresse.

Exercice 2:

Soit la structure suivante

```
-----  
struct Record {  
    char *prenom;  
    char *nom;  
    char *phoneno;  
    address adres;  
};  
-----
```

- Écrire une fonction `Record createRecord(Address ad)` pour la création d'un record.
- Écrire une fonction `affichageRecord(Record *rec)` qui affiche le contenu d'un record. La fonction `affichageAddress(Address *ad)` doit être appelée à l'intérieur de `affichageRecord(Record *rec)`.

Exercice 3: PhoneBook

Soit la structure suivante pour représenter un node d'une liste chaînées

```
-----  
struct PhoneBook {  
    record data;  
    struct PhoneBook *next;  
};  
-----
```

- Écrire une fonction `phonebook* init_phonebook(void)` pour initialiser une liste

- pas NULL.
- Écrire une fonction `void insertRecord(phonebook** phbook)` qui insère un record à la première position de la liste. Utilisez les fonctions `createAddress` et `createRecord` à l'intérieur du corps de la fonction `insertRecord`.
- Écrire une fonction qui affiche tous les éléments de `PhoneBook`.
- Écrire une fonction `void deleteRecord(phonebook** phbook)` pour supprimer la première élément de la liste.
- Écrire une fonction pour calculer la longueur(nombre d'éléments) de `PhoneBook`.

Exercice 4:

Écrire une fonction qui trie le `phonebook` en utilisant le tri par insertion.

Les listes doublement chaînées en C

```
-----  
struct PhoneBook {  
    record data;  
    struct PhoneBook *next;  
    struct PhoneBook *previous;  
};  
-----
```

Exercice 5:

Répétez toutes les exercices pour les listes doublement chaînées.