**Université Mohammed V-Agdal**
**Faculté des Sciences de Rabat**
**Département d'Informatique**

**Année Universitaire 2013/2014**
**MODULE M14 (SMI-S4)**

# *Travaux Pratiques de Structures de Données*
# *[TP n°2 : Types Vecteur & Pile]*

***Objectifs :*** *- Initier à l'implémentation de structures de données par la programmation modulaire ;*
*- Applications de la structure de données Pile.*

***Partie I :*** On désire manipuler des vecteurs de nombres entiers. Pour cela, ***on représente un vecteur par une structure regroupant un entier, pour la taille, et un pointeur sur les éléments du vecteur***. On muni un vecteur des opérations suivantes :

- **`vect`** : *permet de créer un vecteur d'une taille donnée* ;
- **`changer_ième`** : *permet de modifier la composante en position i dans un vecteur* ;
- **`ième`** : *permet d'accéder à la composante en position i dans un vecteur* ;
- **`taille`** : *donne la taille d'un vecteur* ;
- **`lire_vect`** : *permet de saisir un vecteur* ;
- **`afficher_vect`** : *permet d'afficher les composantes d'un vecteur.*

**I.1-** En utilisant la programmation modulaire, implémenter en ***C*** le type **`Vecteur`**.

***Indications :***

- Ouvrir un nouveau projet ***C***, dans l'environnement de programmation ***Dev-C++*** ;
- Créer le fichier "***vect.h***", appelé ***l'interface***, et l'ajouter au projet. Dans ce fichier, placer la déclaration complète, en ***C***, de la structure de données représentant le type **`Vecteur`** ;
- Créer le fichier "***vect.c***", appelé ***la réalisation***, et l'ajouter au projet. Dans ce fichier, définir en ***C*** les opérations fournies par la structure de données. Il est à noter que ce fichier contient au début l'inclusion du fichier "***vect.h***" ;
- Renommer le fichier "***main.c***" par "***testvect.c***", appelé ***l'utilisation***. Dans ce fichier définir une fonction **`main`** pour tester les différentes opérations fournies par la structure de données **`Vecteur`**. Ce fichier doit aussi inclure au début "***vect.h***".

**I.2-** Ajouter et tester les opérations suivantes :

- **`max_vect`** : *permet de retourner la plus grande des composantes d'un vecteur* ;
- **`sum_p`** : *permet de retourner la somme des composantes, de rang pair, d'un vecteur* ;
- **`decalage_circulaire`** : *permet de décaler tous les éléments d'un vecteur d'une place vers la droite (le dernier est inséré à la 1ère place) ;*
- **`tri_vect`** : *permet de trier, dans l'ordre croissant, les composantes d'un vecteur* ;

***Partie II*** *:* Il s'agit d'implémenter et de manipuler une pile de caractères, en utilisant la représentation contiguë d'une pile. ***Une pile est représentée par une structure regroupant un entier, pour le sommet, et un tableau de*** **`MAX_PILE`** ***éléments*** (*ici, des caractères*).

**II.1-** Après avoir implémenté en ***C*** la structure de données **`Pile`** pour les caractères, programmer une fonction **`main`** pour tester une pile caractères. Le programme affiche à l'utilisateur un menu avec les options suivantes :

- ***empiler*** un caractère dans la pile ;
- ***dépiler*** un caractère de la pile ;
- ***taille*** *de la pile ;*
- ***sommet*** *de la pile ;*
- ***afficher*** le contenu de la pile ;
- ***quitter*** le programme.

**II.2-** ***Applications de la structure de données*** **`Pile`** **:**

**a)-** Ecrire un programme ***C*** pour ***inverser une chaîne de caractères***.

**b)-** Ecrire un programme ***C*** permettant de ***vérifier si une chaîne de caractères*** est un palindrome. Un palindrome se lit de la même manière de gauche à droite ou de droite à gauche. **Par exemple**, la chaîne "ressasser" est un palindrome

**c)-** Ecrire un programme ***C*** permettant de ***vérifier si une expression arithmétique contenant les délimiteurs*** **`[`**, **`}`**, **`)`**, **`{`**, **`(`** et **`]`**, ***est correcte ou non***. ***Une expression arithmétique est correcte*** si à chaque délimiteur ouvrant "**`[`**, **`(`**, **`{`**" correspond un délimiteur fermant "**`]`**, **`)`**, **`}`**" de même type. Le dernier symbole ouvert doit être le premier à fermer. Ce qui peut être simulé à l'aide d'une pile. A chaque rencontre d'un délimiteur ouvrant on l'empile et à chaque rencontre d'un délimiteur fermant on dépile.

***Exemples*** *:*

**`{x + (100 – [a + b]) – d}`** est une expression correcte.

**`{x + (y – [a + b]) * 20 – [d + c}`** est une expression incorrecte.