Travaux Dirigés de Programmation Impérative n°7

Cours d'informatique de Première Année

—Licence MIEF/SPI—

Liste, Pile et File

Nous avons vu en cours la notion de liste chaînée. En la modifiant légèrement, on va voir qu'une même structure permet de coder les notions de pile et de file.

Au fur et à mesure que vous écrirez les fonctions, il est conseillé de tester qu'elle fonctionne en écrivant un petit programme qui les utilise.

▸ Exercice 1. Liste chainée

– *Soit les structures suivantes :*

```
struct maillon_s{
int val;
struct maillon_s * suivant;
};
typedef struct maillon_s maillon_t;

struct liste_s{
struct maillon_s * premier;
struct maillon_s * dernier;
int taille;
};
typedef struct liste_s liste_t;
```

– *Écrire la fonction* `liste_initialiser` *qui crée une liste vide.*
– *Écrire la fonction* `liste_vide` *qui renvoit* 1 *si la liste est vide et* 0 *sinon.*
– *Écrire la fonction* `liste_cardinal` *qui renvoit le nombre d'entiers stockés dans la liste.*
– *Écrire la fonction* `liste_ajouter_debut` *qui prends en entrée un pointeur sur une* `liste_t` *et un entier* x *et ajoute* x *au debut de la liste.*
– *Écrire la fonction* `liste_ajouter_fin` *qui prends en entrée un pointeur sur une* `liste_t` *et un entier* x *et ajoute* x *à la fin de la liste.*
– *Écrire la fonction* `liste_extraire_debut` *qui prends en entrée un pointeur sur une* `liste_t` *extrait le maillon qui se trouve au début de la liste et le renvoit.*
– *Écrire la fonction* `liste_afficher` *qui prends en entrée une* `liste_t` *et affiche son contenu.*
– *Écrire la fonction* `liste_appartient` *qui prends en entrée une* `liste_t` *et un entier* x *et renvoit* 1 *si* x *appartient à la liste et* 0 *sinon.*
– *Écrire la fonction* `liste_supprimer` *qui prends en entrée un pointeur sur une* `liste_t` *et un entier* x*. La fonction supprime tous les maillons qui contiennent la valeur* x*.*
– *Écrire la fonction* `liste_detruire` *qui prends en entrée un pointeur sur une* `liste_t` *et désalloue l'espace mémoire qu'elle occupe.*

– *Quelle est la complexité en temps et en espace de toutes ces fonctions ?*

▸ **Exercice 2. Pile et File**

En réutilisant la notion de liste, définir les notions de pile et de file. On créera les différentes notions dans différents fichiers. Puis, utiliser les fonctions qui manipulent les listes pour définir les fonctions suivantes sur les piles et les files :

– *les fonctions* `pile_initialiser` *et* `file_initialiser`,
– *les fonctions* `pile_vide` *et* `file_vide`,
– *les fonctions* `pile_hauteur` *et* `file_longueur`,
– *la fonction* `enfiler` *qui prends en entrée un pointeur sur une* `file_t` *et un entier* x *et ajoute* x *à la fin de la file.*
– *la fonction* `empiler` *qui prends en entrée un pointeur sur une* `pile_t` *et un entier* x *et ajoute* x *au début de la pile.*
– *la fonction* `defiler` *qui prends en entrée un pointeur sur une* `file_t`, *extrait l'élément qui se trouve au début et le renvoit.*
– *la fonction* `depiler` *qui prends en entrée un pointeur sur une* `pile_t`, *extrait l'élément qui se trouve au début et le renvoit.*
– *les fonctions* `pile_détruire` *et* `file_détruire`,