Travaux Pratiques d'algorithmique n°7

Cours d'Informatique de Deuxième Année

—DEUG Sciences M3-MI—

Listes Chaînées

Le but de ce TP est de manipuler les listes chaînées au travers de l'implémentation d'une pile. La notion de pile doit être vue comme un type abstrait. Ainsi, toute opération sur un objet de type pile ne doit pas laisser apparaître la structure sous-jacente. De plus, en considérant ce travail comme la création d'une bibliothèque de fonctions celles-ci doivent toutes retourner un code d'erreur en cas de problème (cf `scanf`). La fin de ce TP sera consacré à un cas particulier : l'implémentation d'une calculatrice de type HP.

La pile

La structure de données pour la pile est la suivante :

```
struct Cellule_struct {
  int entier;
  struct Cellule_struct *suivant;
};

typedef struct Cellule_struct Cellule;

typedef Cellule * Pile;
```

▸ **Exercice 1.** *Ecrire les fonctions :*

- `Cellule *CreerCellule(int entier, Cellule *suivant)`
- `void DetruireCellule(Cellule *cellule)`

pour respectivement, allouer de la mémoire pour une cellule contenant les paramètres en arguments et libérer la mémoire prise par une cellule. Pour la fonction `Cellule *CreerCellule(int entier, Cellule *suivant)`*, la valeur de retour sera* `NULL` *en cas de manque d'espace mémoire.*

Ces fonctions sont des fonctions de "bas niveau" qui ne sont logiquement pas utilisées dans le programme principal.

▸ **Exercice 2. (manipulation)** *Ecrire les fonctions :*

- `int InitialiserPile(Pile *pile)`,
- `int EstVidePile(Pile pile)`,
- `int EmpilerPile(Pile *pile, int entier)`,
- `int DepilerPile(Pile *pile, int *entier)`,
- `int SommetPile(Pile pile, int *entier)`,

permettant respectivement, d'initialiser une pile, de tester si une pile est vide, d'empiler un entier sur la pile, de dépiler un entier de la pile, et de retourner le sommet de la pile. Ces fonctions retournent 0 en cas de problème et 1 sinon.

▸ **Exercice 3.** *Ecrire la fonction* `int ViderPile(Pile *pile)` *permettant de libérer l'espace mémoire occupé par une pile.*

▸ **Exercice 4.** *(***Affichage***)* *Ecrire la fonction* `void AfficherPile(Pile pile)` *permettant d'afficher la pile.*

▸ **Exercice 5.** *(***Application***)* *Réécrire la fonction factorielle en simulant la récursivité à l'aide d'une pile. Que pensez-vous de cette implémentation ?*

▸ **Exercice 6. (la calculette)** *Ecrire la fonction* `int Calculatrice(void)` *qui implémente une calculatrice de type HP et renvoie le dernier résultat des calculs. On implémentera les fonctions suivantes :*

- *? : affiche l'aide*
- *p : affiche la pile*
- *+,-,/,* : dépile deux nombres de la pile, effectue l'opération et empile le résultat*
- *0..9 : empile le chiffre sur la pile*
- *v : vide la pile*
- *q : quitte*