Travaux Pratiques d'algorithmique n°8

Cours d'Informatique de Deuxième Année

—DEUG Sciences S3-MI—

Pile

Le but de ce TP est de manipuler une *pile* d'**entiers**. Cette 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.
Pour illustrer ceci, nous allons créer deux bibliothèques de fonctions implémentants les opérations de base sur les piles.
Nous utiliserons ces bibliothèques pour implémenter une calculatrice de type HP.

---

1 Représentation par un tableau

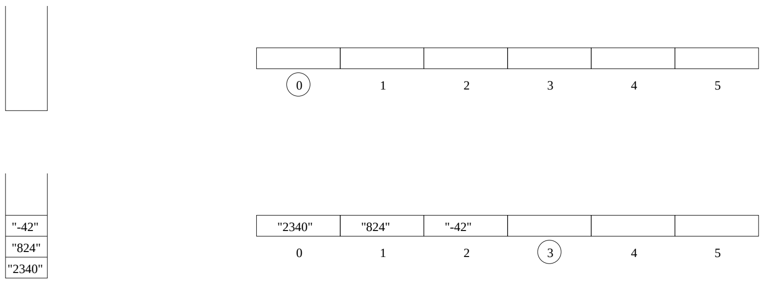
Dans cette première section, la pile est codée par la structure suivante :

```
#define MAXPILE 30

struct pile
{
  int Tab[MAXPILE];
  int Hauteur;
} ;

typedef struct pile Pile;
```

Ainsi, nous avons les représentations suivantes :



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

- *void InitialiserPile(Pile *p)*,

- *`int EstVidePile(Pile p)`*,
- *`int EmpilerPile(Pile *p, int val)`*,
- *`int DepilerPile(Pile *p, int *val)`*,
- *`int SommetPile(Pile p, int *val)`*,
- *`void ViderPile(Pile *p)`*,

permettant respectivement, d'initialiser une pile, de tester si une pile est vide, d'empiler un entier sur une pile, de dépiler un entier d'une pile, de retourner l'entier située au sommet d'une pile, et de vider une pile.
Certaines fonctions retournent 0 en cas de problème et 1 sinon.

▸ **Exercice 2.** ***(Affichage)*** *Écrire une fonction* `void AfficherPile(Pile p)` *permettant d'afficher le contenu de la pile* `p`*.*

2 Application

▸ **Exercice 3.** ***(Calculette simple)*** *Écrire une fonction* `int Calculatrice(void)` *qui implémente une calculatrice de type HP (cf la commande* `dc`*) et renvoie le dernier résultat des calculs.*
On vous demande d'implémenter les fonctionnalités de base :

- *empiler un chiffre (*`0`*,* `1`*, …,* `9`*) ;*
- *appliquer un opérateur simple (*`+`*,* `-`*,* `*`*,* `/`*) ; (dépiler deux fois, et empiler le résultat) ;*
- *afficher le sommet de la pile (*`s`*) ;*
- *afficher le contenu de la pile (*`p`*) ;*
- *quitter (*`q`*).*

3 Représentation par une liste chaînée

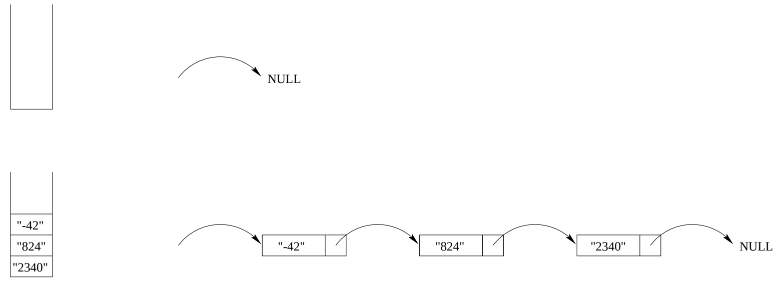
Dans cette section, la pile est codée par une structure chaînée :

```
struct cell_struct
{
  int                  Entier;
  struct cell_struct * Suiv;
};

typedef struct cell_struct Cellule;

typedef Cellule * Pile;
```

Ainsi, nous avons les repŕesentations suivantes :



▸ **Exercice 4.** *L'objectif de cet exercice est de réécrire toutes les fonctions de manipulation de la pile des exercices 1 et 2.*

4 Application (suite)

▸ **Exercice 5.** *Si vous avez tenu compte des spécifications demandées pour la programmation de toutes ces fonctions, la calculatrice que vous avez conçue (cf exercice 3) doit fonctionner sans aucune modification à l'aide de cette nouvelle représentation par liste chaînée. Tester si c'est bien le cas...*

▸ **Exercice 6.** ***(Calculatrice "évoluée")*** *On vous demande maintenant d'implémenter l'ensemble des fonctionnalités suivantes :*

- *? : affiche l'aide ;*
- `nb` *: empile l'**entier*** `nb` *;*
- `+`, `-`, `/`, `*` *: effectue l'opération associée ;*
- `s` *: affiche le sommet de la pile ;*
- `p` *: affiche le contenu de la pile ;*
- `v` *: vide la pile ;*
- `q` *: quitte.*

Indication : le programme suivant permet la saisie de (plusieurs) commandes. Chaque commande est séparée par un espace ou par un retour chariot.

```
#define MAXFLUX 200

void main(void)
{
  char *line;   /* ligne des commandes */
  char *com;    /* commande courante   */
```

```
  int   i;       /* compteur            */
  int   flag=1;  /* on continue ?       */

  /* Allocation de la chaine de caracteres pour la ligne des commandes */
  line=(char*)malloc(MAXFLUX*sizeof(char));

  while ( flag == 1 )
  {
    /* saisie de la ligne des commandes */
    printf("# ");
    i=0;
    while ( ( i < MAXFLUX ) && ( (line[i]=getchar()) !='\n' ) )
          i++;
    line[i] = '\0';

    com=(char*)strtok(line," ");
    while ( ( com != NULL ) && ( flag == 1 ) )
          {
            printf("%s\n",com);
            /* ...................... */

            if ( com[0] == 'q' )
               flag = 0;

            /* commande suivante: */
            com=(char*)strtok(NULL," ");
          }
  }
  free(line);
}
```

Il sera nécessaire de programmer les fonctions :

- `int IsInteger(char *ch)` *qui teste si la chaîne* `ch` *code un entier ;*
- `int ToInt(char *ch)` *qui retourne l'entier codé par la chaîne* `ch` *;*
- `int IsOp(char *ch)` *qui teste si la chaîne* `ch` *code un opérateur (+,-,/,*).*