

Structures de données et algorithmes

TP : Implantation contiguë et chaînée des piles

Avant de commencer

On rappelle la signature de la spécification algébrique des piles vue en cours.

Spéc PILE(TRIV) étend BASE

Sorte Pile

Opérations

```
pilenouv :      -> Pile
empiler  : Pile S -> Pile
depiler  : Pile  -> Pile
remplacer : Pile S -> Pile
sommet   : Pile  -> S
vide     : Pile  -> Bool
hauteur  : Pile  -> Nat
```

1 Gestion contiguë

On considère des piles de caractères C et l'implantation contiguë avec la structure de données:

```
#define MAX_P 50
typedef struct {
    char tab[MAX_P];
    Nat h;
} Pile;
```

Écrivez proprement le fichier header correspondant, de nom `pile_char.h` avec les structures de données et les profils des fonctions, puis le fichier contenant le source des fonctions, de nom `pile_char.c`.

Écrivez dans un fichier de nom `test.c` un programme permettant de tester toutes les opérations sur les piles de caractères. Écrivez également le fichier `Makefile` permettant de faire la compilation séparée de ces fichiers.

Programmez dans un fichier de nom `parentheses.c` un programme permettant de tester si une chaîne de caractères est bien parenthésée. Les parenthèses considérées seront les suivantes : `([{}])`.

2 Gestion chaînée

Même exercice qu'à la section précédente mais en utilisant une implantation chaînée comme celle vue en cours. On considérera donc la structure de données :

```
typedef struct spile {
    char v;
    struct spile *s;
} Spile, *Pile;
```

Les noms des fichiers seront respectivement `pile_char2.h`, `pile_char2.c`. Vous vous arrangerez pour que le source des programmes de test, écrits dans `test.c` et `parentheses.c`, reste inchangé.