Université de Strasbourg
UFR de mathématique et d'informatique

L2 informatique
automne 2015

Structures de données et algorithmes

TP4 : Implantation chaînée des piles

Avant de commencer

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

```
Spéc PILE(TRIV) etend BASE
Sorte Pile
Opérations
    pile_vide :          -> Pile
    emmpiler  : Pile  S -> Pile
    depiler   : Pile    -> Pile
    remplacer : Pile  S -> Pile
    sommet    : Pile    -> S
    vide      : Pile    -> Bool
    hauteur   : Pile    -> Nat
```

Gestion contiguë

On considère des piles de caractères C et l'implantation contiguë vue en cours 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`, puis le fichier contenant le source des fonctions, de nom `pile_char.c`.

Écrivez dans un fichier de nom `test.c` un petit 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 `parenthses1.c` un programme permettant de tester si une chaîne de caractères est bien parenthésée. Les parenthèses considérées seront les suivantes : `([{}])`.

1 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 `parenthses1.c`, reste inchangé.