Programmation C
TP 4

Une pile d'entiers C statique avec un tableau.

Une pile est une structure de donnée informatique collectant des objets tout en gardant leur ordre d'apparition. Dans une pile, le premier à pouvoir sortir est toujours le dernier à être rentré.

Nous allons faire des piles d'entiers (utilisant le type C `int`) et statiques (gestion automatique de la mémoire et taille finie).

Comme votre module de pile (composé de deux fichiers `stack.h` et `stack.c`) ne devra manipuler qu'une seule pile, il suffira d'utiliser une variable globale statique dans le fichier source de votre module. Cette unique pile sera ainsi confinée dans un seul fichier et l'utilisateur n'aura pas à se soucier de la mémoire.

Exercice 1 **Un module pour des piles**

Attention, le nom des fonctions est très important! Ne changez pas ces noms!

Implanter un module de pile manipulant une seule pile de taille 15 au maximum.

Nous donnons le fichier des entêtes de ce module (téléchargeable):

fichier `stack.h`

```
1  #ifndef _STACK_
2  #define _STACK_
3
4  #define MAX_SIZE 15
5
6  typedef struct stack{
7    int values[MAX_SIZE];
8    int size;
9  }Stack;
10
11 /* Initialize correctly the stack */
12 void stack_init(void);
13
14 /* Returns the current size of the stack. */
15 int stack_size(void);
16
17 /* Returns 1 if the stack is empty, returns 0 otherwise. */
18 int stack_is_empty(void);
19
20 /* Returns the element at the top of the stack. */
21 int stack_top(void);
```

```
22
23 /* Pops the element at the top of the stack and returns it. */
24 int stack_pop(void);
25
26 /* Pushs a given integer 'n' at the top of the stack. */
27 void stack_push(int n);
28
29 /* Displays the content of the stack on the standard output. */
30 void stack_display(void);
31
32 /* Returns the element at index 'index' inside the stack. The user is
33    responsible of the use of this function. The answers may not be
34    relevant if a call is done outside the current size of the
35    stack. */
36 int stack_get_element(int index);
37
38 #endif
```

Vous devez maintenant implanter le fichier de code qui devrait commencer comme il suit :

fichier `stack.c`

```
1  #include "stack.h"
2
3  /* This module proposes a single global and static stack  */
4  static Stack stack;
5
6  /* Initialize correctly the stack */
7  void stack_init(void){
8    stack.size = ...
9  }
10
11 ...
12 ...
```

Faites un fichier `main.c` qui importe le fichier `stack.h` et testez chacune de vos fonctions dans un `main`. Il s'agit de partir d'une pile vide correctement initialisée et d'imprimer votre pile à l'écran entre chaque opération.

Exercice 2 **Branchement de votre module de pile**

Testez la fonctionnalité `dc` de UNIX. C'est une calculatrice non graphique en polonais inversé. `man dc` vous dira comment utiliser rapidement cet utilitaire.

Communément appelées polonais inversé, ces notations portent aussi le nom plus compliqué d'expressions algébriques postfixées. Cela signifie que les opérations se placent en fin d'expression et portent sur les nombres précédents (qui sont rentrés dans une pile). L'expression infixée `1 + 2` devient `1 2 +` en notation postfixée.

Téléchargez le bout de code `calc.tar.gz` pour ce tp4. Décompressez et regardez le contenu de l'archive et lisez le `makefile`.

Branchez votre module de pile (donc les deux fichiers) dans ce bout de projet, compilez et testez l'exécutable généré. Gérez les conflits de branchement s'il y en a.

Pour les étudiants utilisant leur ordinateur personnel, la bibliothèque graphique `libMLV` est nécessaire pour la compilation de ce code (les ordinateurs de la fac sont équipés de cette bibliothèque). On peut trouver des instructions d'installation à l'adresse :
http://www-igm.univ-mlv.fr/~boussica/mlv/api/French/html/index.html
ou simplement faire une recherche `google` sur le mot clé libMLV. C'est cette même bibliothèque qu'il faudra utiliser pour faire la partie graphique de votre projet durant le second semestre.

Exercice 3 Petits raffinements d'une calculatrice en polonais inversée

Une fois votre module de pile correctement branché dans ce projet, tentez de rajouter les fonctionnalités suivantes :

- le calcul des modulos avec le `char` '%', reste de la division euclidienne dans les entiers (opération binaire : deux arguments)
- le calcul de la factorielle avec le `char` '!' (opération unaire : un seul argument).