Licence Informatique **TD Algorithmique 3**
Université Bordeaux 1

Pile

Fichiers sources : `http://dept-info.labri.u-bordeaux.fr/ENSEIGNEMENT/algo3`

« Une pile est un type abstrait particulièrement simple, mais souvent très utile. C'est un conteneur avec une restriction assez sévère sur l'accès des objets contenus. L'idée principale de la pile est que les objets sont récupérés à l'ordre inverse par rapport a l'ordre dont ils ont été insérés. Cette restriction permet des implémentations très efficaces. » [R. Strandh]

Signature. Les opérations possibles sur une pile sont :

- empiler un objet quelconque ;
- dépiler le dernier objet empilé ;
- récupérer l'objet au sommet de la pile ;
- déterminer si la pile est vide.

Interface. Voilà l'interface C que nous proposons pour une pile (fichier `stack.h`).

```
struct stack; typedef struct stack *stack;

/* create an empty stack */ extern stack stack_create(void);

/* push an object on a stack */ extern void stack_push(stack s, void
*object);

/* return true if and only if the stack is empty */ extern int
stack_empty(stack s);

/* return the top element of the stack. */ extern void
*stack_top(stack s);

/* pop an element off of the stack. */ extern void stack_pop(stack
s);
```

Exercice 1. - Implantation d'une pile avec un tableau extensible L'objectif de cet exercice est d'implanter l'interface proposée en utilisant un tableau extensible .

1. On peut imaginer différente stratégie d'extension du tableau lorsque ce-dernier devient trop petit pour empiler un nouvel élément. On choisira d'abord la stratégie consistant à étendre la taille du tableau lorsqu'il est plein (ajouter M nouvelles entrées).
   Vous décrirez d'abord votre implémentation en langage pseudo-code, sans référence aux particularités du langage C. Pour simplifier, vous supposerez en particulier qu'un tableau connaît sa taille (tableau.taille).
2. Donnez ensuite une réalisation en C de cette implémentation.(Le fichier `stack_array.c` propose une solution). Tester votre implantation avec le programme de test fourni `test_stack.c`).

3. Générer des cas tests de tailles n croissantes et mesurer le temps avec la fonction `gettimeofday()`. Afficher graphiquement le coût amortie à l'aide de GNUPlot. Vous pouvez utiliser pour cela les fichiers `ALIRE`, `resultat.plot`, `perf_stack.c` fournis dans l'archive.
4. On peut aussi évaluer le temps requis en moyenne pour l'insertion d'un élément de manière théorique. Vous comparerez ce résultat avec les exéprimentations).

   Vous supposerez que votre implémentation utilise un tableau de taille M. L'insertion du N ème élement dépend donc de la valeur de N modulo M (on procède à l'extension du tableau et à la recopie des valeurs lorsque $N = k \cdot M + 1$ est un multiple k plus 1).
5. Au vu du calcul précédent, étudiez d'autres stratégies d'extension. Vaut-il mieux n'étendre le tableau que de $M/2$ case de plus ? Quel effet cela a t-il sur le temps moyen d'insertion d'un nouvel élement ? Etudiez la stratégie consistant à *doubler* la taille du tableau lorsque le tableau est plein.

Exercice 2. - Implantation d'une pile avec une liste chaînée

L'objectif de cet exercice est d'implanter l'interface proposée en utilisant une liste chaînée.

1. Vous décrirez d'abord votre implémentation en langage pseudo-code, sans référence aux particularités du langage C.

   Discutez des avantages et inconvénients (par rapport à l'implémentation tableau de l'exerccie 1).
2. Donnez ensuite une réalisation en C de cette implémentation. (Le fichier `stack_list.c`) propose une solution). Tester votre implantation avec le programme de test fourni `test_stack.c`).
3. Générer des cas tests de tailles n croissantes et mesurer le temps avec la fonction `gettimeofday()`. Afficher graphiquement le coût amortie à l'aide de GNUPlot. Vous pouvez utiliser pour cela les fichiers `ALIRE`, `resultat.plot`, `perf_stack.c` fournis dans l'archive.

Exercice 3. - Implantation d'une pile avec une "liste chaînée de tableaux"

Bonus. Réfléchir à une nouvelle implémentation à base d'une liste chaînée de tableaux de tailles fixes. (On ajoute un nouveau tableau à la liste au besoin.)

Discutez des avantages et inconvénients par rapport aux deux implémentations précédentes.