Compte-rendu TP2:
Gestion d'une pile et d'une file.

I°) Sujet du tp:

- Traduire en langage C les procédures et fonctions de gestion de pile vues en cours, pour une pile dont le type d'éléments est défini par typedef
- Traduire en langage C les procédures et fonctions de gestion de files vues en cours, pour une file d'entiers
- Pour tester le bon fonctionnement de ces deux structures, écrire le programme d'inversion d'une pile traité dans le partiel 2.

choix de programmation:

Pour une question pratique le choix de type de pile ou de file se fera par un define au lieu d'un typedef.

II°)Code source commenté:

Fourni en fin de compte rendu.

III°)Compilation.

Pour les traces d'exécution, on a utilisé Débian 2.6 (LINUX):

```
gcc -c file.c -Wall -ansi -pedantic
gcc -c pile.c -Wall -ansi -pedantic
gcc -c pilefile.c -Wall -ansi -pedantic
gcc file.o pile.o pilefile.o -o tp2 -Wall -ansi -pedantic
```

Le code produit respecte donc la norme ansi du C

IV°)Trace & cas particulier:

Pour la première partie on travaillera sur les fonctions de la pile.

Init:

```
Fonctions de test de la pile
1) Initialisation de la pile
2) test si pile vide
3) test si pile pleine
4) empiler une valeur
5) depiler une valeur
6) valeur du sommet
autre: retour au menu
1
taille maximale souhaitee?
10
/*aucun message d'erreur*/
```

Sommet:

cas normal:

```
la valeur a empiler?   /*on montre la dernière empilation*/
2
empilation reussi

valeur du sommet : 2
```

cas pile vide

```
/*on effectue l'appel de la fonction sommet juste après l'initialisation donc la pile est bien vide*/
sommet non atteignable
```

Pleine:

cas pile pleine:

```
/*on initialise une pile de 3 éléments et on empile tant que cela réussi */
la valeur a empiler?
4
empilation rate

pile pleine
```

cas pile non-pleine:

```
/*on reprend l'exemple précédent en dépilant une valeur:*/
valeur depile : 3

pile non pleine
```

Vide:

cas pile vide:
/*on effectue l'appel de la fonction vide juste après l'initialisation donc la pile est bien vide*/
pile vide

cas pile non vide:
/*on reprend le contexte précédent et on empile une valeur*/
la valeur a empiler?
1

pile non vide

Empiler:

cas normal:
la valeur a empiler?
10
empilation reussi

cas pile pleine:
/*on empile tant que la pile n'est pas pleine*/
la valeur a empiler?
5
empilation rate

Dépiler:

cas normal:
/*on reprend le cas précédent (pile pleine) dernier valeur empiler : 5 */
valeur depile : 5

cas pile vide:
/*on effectue l'appel de la fonction depiler juste après l'initialisation donc la pile est bien vide*/
depilation rate

Pour la deuxième partie on travaillera sur les fonctions de la file.

Init:

Fonctions de test de la file
1) Initialisation de la file
2) test si file vide
3) test si file pleine
4) enfiler une valeur
5) defiler une valeur
6) valeur du sommet
autre: retour au menu
1
taille maximale souhaitee?
10
/*aucune erreur détecté*/

Sommet:

cas normal:
/*on montre le dernier enfilage*/
la valeur a enfiler?
3
enfilage reussi

valeur du sommet : 3

cas file vide:
/*on effectue l'appel de la fonction sommet juste après l'initialisation donc la file est bien vide*/
sommet non atteignable

Pleine:
cas file pleine:
/*On enfile tant que cela réussit*/
file pleine

cas file non-pleine:
/*on reprend le contexte précédent où on défile une valeur*/
file non pleine

Vide:
cas file vide:
/*on effectue l'appel de la fonction vide après l'initialisation donc la file est bien vide*/
file vide

cas file non vide:
/*On reprend le contexte précédent auxquels on enfile une valeur*/
file non vide

Enfiler:

cas normal:

/*on effectue l'appel de la fonction vide après l'initialisation donc la file est bien vide*/
la valeur a enfiler?
3
enfilage réussi

cas file pleine:

/*on enfile tant que la file n'est pas pleine*/
/*on enfile une dernier valeur*/
la valeur a enfiler?
5
enfilation rate

Défiler:

cas normal:

/*on se place dans le contexte précédent (pile pleine)*/
/*première valeur enfiler*/
la valeur a enfiler?
2
enfilage reussi

valeur defile : 2

cas file vide:

/*on défile tant que la file n'est pas vide*/
defilage rate

Maintenant on effectue le test sur l'exercice de l'examen:

Combien d'element doit contenir la pile initiale
5
initialisation des piles
remplissage de la pile d'origine
valeur?
1
valeur?
2
valeur?
3
valeur?
4
valeur?
5 /* on a empilé le 1 puis le 2,3,4,5*/
on depile dans la file
on defile dans la pile
on depile toute la pile

1 2 3 4 5 /*on a dépilé le 1 puis le 2,3,4,5 */
destruction des files et des piles /*on a donc bien le fonctionnement attendu*/

V°) Test libération de la mémoire: Valgrind

Pour cela on a utilisé au minimum une fois toutes les fonctions crées.

==3842==
==3842== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 13 from 1)
==3842== malloc/free: in use at exit: 0 bytes in 0 blocks.
==3842== malloc/free: 8 allocs, 8 frees, 148 bytes allocated.
==3842== For counts of detected errors, rerun with: -v
==3842== All heap blocks were freed -- no leaks are possible.

Donc aucun memory leak (fuite de mémoire)