Algorithmique et structures de données 2
TD/TP1

am@up8.edu

2022

Dans ce TD :
— Correction de code et optimisation
— Conception petite fonction

1 Exercice 0 : Avant toute chose

— rejoignez le canal Mattermost L2-X Algorithmique et structures de données 2 (2022-2023) `https://talk.up8.edu/`
— envoyez moi un mail ayant pour objet "[ASD2]" depuis le mail que vous souhaitez utiliser (si pas de réponse au bout de 48h —hors week-end—, me relancer)
— inscrivez-vous au cours "Licence Informatique (L2-X) Algorithmique et structures de données 2" mot de passe ALGODONNEES
— inscrivez-vous dans un groupe pour la création de la *mindmap* (lien sur le moodle)

2 Exercice 1 : Tri à bulle (sans coder)

Question 1 : Pour le tableau en entrée : `T[5] = {8,1,4,3,1}`, l'algorithme suivant de tri à bulle donne le résultat `8 1 3 4 1`.
Corrigez la fonction :

```
1  int tri_a_bulle(int T[], int n)
2  {
3      int temp;
4      for (int i=n-1 ; i > 0 ; i--){
5          for(int j = 1 ; j < i-1 ; j++) {
6              if(T[j] > T[j+1]){
7                  temp = T[j+1];
8                  T[j+1] = T[j];
9                  T[j] = temp;
10             }
11         }
12     }
13     return 0;
```

```
14 }
```

Indice : regarder quelle sont les plages de valeurs couvertes par `i`, `j` et `i+1`

Question 2 : Pour un tableau de 5 valeurs déjà triées, combien d'itérations sont effectuées ? Comment optimiser cette fonction afin de limiter les itérations inutiles ? (3-4 lignes de code à rajouter)

Question 3 : Les lignes 7 à 9 correspondent à une permutation de valeurs. Écrivez une fonction permute et l'appel correspondant permettant de remplacer ces trois lignes.

```
1 int permute(           ,           ){
2
3
4
5     return 0;
6 }
```

Appel correspondant :

```
1 permute(           ,           );
```

3 Types de variables (sans coder)

Question 1 On souhaite qu'un appel à la fonction `appels_a_foo` produise l'affichage suivant :

```
Appel à foo() numéro 1
Appel à foo() numéro 2
Appel à foo() numéro 3
```

Complétez les lignes 2 et 3 afin d'obtenir le bon affichage.

```
1 int foo(void){
2
3
4         printf("Appel à foo() numéro %d", cpt);
5         return 0;
6 }
7
8 int appels_a_foo(){
9         int i ;
```

```
10          for (i = 0; i<10; i++){
11                  foo();
12          }
13          return 0;
14  }
```

4 Implémentation

4.1 Manipuler `gdb`

Attention : pour pouvoir utiliser `gdb`, il faut compiler avec l'option `-g`.

Question 1 Testez le bon fonctionnement des fonctions ci-dessus. Entraînez-vous à utiliser `gdb` pour placer des `breakpoints` et afficher les valeurs des variables.

Vous pouvez utiliser la documentation disponible ici : `https://www.rocq.inria.fr/secret/Anne.Canteaut/COURS_C/gdb.html`, ainsi que les formats d'affichage suivants `https://sourceware.org/gdb/onlinedocs/gdb/Output-Formats.html`.

Question 2 Récupérez sur moodle les codes des fonctions vus en cours et utilisez les outils de debuggage présentés pour les corriger / afficher les valeurs.

- `liste.c`, `liste.h` : créer une liste et afficher les éléments qu'elle contient (adresses et valeurs) dans `gdb`.
- testez et corrigez le code `structbug.c`.
- testez et corrigez le code `menu.c` (il faut *vider le tampon d'entrée*).

4.2 Fuites mémoire et outil `valgrind`

- `fuite.c` : menez le diagnostic avec valgrind et corrigez le code.