Université de Lille

# Contrôle TP

R2.04
DUT Info
Julien Baste
2021–2022

---

L'objectif de ce CTP est que vous soyez capable de nous montrer que vous savez utiliser les bases du langage C ainsi que de comprendre comment gérer la mémoire en fonction des structures de données qui vous sont imposées.

Attention le sujet est volontairement long et la capacité à répondre rapidement fait partie intégrante de l'évaluation. Il y a trois exercices. L'exercice 1 correspond à des questions de cours et l'exercice 2 est faisable par tout le monde. L'exercice 3 est accessible mais volontairement plus difficile. À noter qu'il se base sur l'exercice 2.

Information : Nous évaluons votre code sans forcément l'exécuter. De ce fait, si vous n'arrivez pas à écrire certaines fonctions, vous pouvez tout de même les utiliser dans les fonctions suivantes (même si du coup vous ne pourrez pas les tester). Cela peut vous permettre de ne pas rester bloquer et de récupérer quelques points sur des choses que vous savez faire.

Comme annoncé, vous avez le droit à une feuille de papier et un crayon ainsi qu'aux documents présents sur le Moodle du cours, mais pas à votre session.

Vous travaillerez dans un dossier `CTP_R2.04` que vous allez créer pour l'occasion. Vous devez écrire la totalité de votre code dans un fichier `main.c` que vous déposerez sur Moodle pour être évalué. Avant de fermer votre session, assurez-vous que c'est le bon fichier qui a été déposé sur Moodle.

## Exercice 1 : Question de cours (5 points)

**Q1.** (1) Écrivez une fonction `void exemple_pointeur (void);` dans laquelle vous :
- définissez une variable `ma_variable` de type entière,
- initialisez `ma_variable` avec la valeur de votre choix,
- affichez le contenu de `ma_variable`,
- affichez l'adresse mémoire à laquelle se situe `ma_variable`,
- définissez un pointeur `mon_pointeur` qui cible `ma_variable`,
- affichez la valeur de `mon_pointeur`,
- affichez le contenu de la case mémoire ciblée par `mon_pointeur`.

**Q2.** (1) Écrivez une fonction `void exemple_tableau(char* tableau, int taille);` qui prend en argument un tableau que l'on supposera de taille au moins 3 et qui affiche l'adresse mémoire de la case d'indice 2 du tableau puis qui affiche le déréférencement de cette case. Vous devriez afficher quelque chose de la forme :

```
La deuxième case du tableau se situe à l'adresse 0x7ffe837c0544
Elle contient : c
```

**Q3.** (1) Écrivez une fonction `void caractere_suivant (char* pt_car);` qui prend un pointeur vers un caractère et transforme ce caractère ciblé en le caractère suivant selon la table ascii. Ainsi `'a'` deviendra `'b'`, `'2'` deviendra `'3'`, `'*'` deviendra `'+'` etc. On ne se préoccupera pas des cas limites de dépassement de la table ascii.

**Q4.** (1) Écrivez une fonction `int taille_chaine_de_caracteres(char* chaine);` qui prend un tableau de caractère correspondant à une chaine de caractères et qui retourne la taille de cette chaine.

**Q5.** (1) Écrivez une fonction `void affiche_tableau (int* tableau, int taille);` qui affiche le tableau `tableau`. Par exemple, si le tableau contient 3, 6, 8, il s'affichera de la manière suivante : `3 / 6 / 8`

## Exercice 2 : Bouquets de fleurs (7 points)

Un fleuriste veut utiliser une structure de données pour garder en mémoire la composition des bouquets qu'il doit préparer. Le fleuriste est spécialisé en roses, en tulipes et en pivoines. Chaque bouquet considéré pouvant avoir un nombre de fleurs de chaque type différent, il vous sera demandé de créer une structure de donnée `bouquet_t` contenant trois attributs `roses`, `tulipes` et `pivoines`. Chacun de ces attributs est un entier représentant le nombre de fleurs du type désigné.

**Q1.** (1) Définissez le type `bouquet_t` à l'aide d'une structure.

**Q2.** (1) Écrivez une fonction `bouquet_t* nouveau_bouquet (void);` qui réserve la place en mémoire pour stocker un élément de type `bouquet_t` et retourne un pointeur qui cible cet espace mémoire.

**Q3.** (1) Écrivez une fonction `void initialise_bouquet(bouquet_t* bouquet, int nb_roses, int nb_tulipes, int nb_pivoines);` qui initialise le bouquet avec le bon nombre de chaque fleur.

**Q4.** (1) Écrivez une fonction `void affiche_bouquet(bouquet_t* bouquet);` qui affiche le contenu du bouquet de la manière suivante :

```
1 rose, 2 tulipes et 3 pivoines
```

Note : On considérera pour cette question que le bouquet passé en argument existe toujours et que les nombres de roses, tulipes et pivoines sont toujours des nombres positifs.

Attention : il est attendu que l'affichage gère correctement la présence ou non des `'s'` du pluriel. C'est-à-dire qu'il n'y a pas de `'s'` dans le cas où la valeur est de `0` ou `1` et il y en a si la valeur est au moins `2`.

**Q5.** (1) Écrivez une fonction `void libere_bouquet(bouquet_t* bouquet);` qui libère l'espace mémoire alloué pour le bouquet.

**Q6.** (2) Écrivez une fonction `bouquet_t* combine_bouquets(bouquet_t* bouquet_1, bouquet_t* bouquet_2);` qui prend en paramètre deux bouquets et en crée un troisième dont le nombre de fleurs de chaque type est la somme des fleurs de ce type dans `bouquet_1` et `bouquet_2`. Les deux bouquets donnés en argument seront supprimés par la fonction.

## Exercice 3 : Commandes de fleurs (9 points)

Notre fleuriste étant spécialiste en décoration de mariage, on lui commande systématiquement plusieurs bouquets en même temps. Afin de travailler avec les commandes qui lui sont adressées, vous allez définir une structure `commande_t` qui contiendra deux attributs, le premier, `nb_bouquets`, est un entier qui stockera le nombre de bouquets de la commande et le deuxième, `tableau`, est un tableau dont les éléments seront de type `bouquets_t*`. Chaque élément du tableau est donc un pointeur vers un bouquet.

Pour cet exercice, vous pouvez réutiliser les fonctions que vous avez écrites dans l'exercice précédent. Cela est même vivement conseillé pour certaines questions.

**Q1.** (1) Définissez le type `commande_t` à l'aide d'une structure.

Indice : Un tableau d'éléments de type `bouquet_t*` est de type `bouquet_t**`

**Q2.** (2) Écrivez une fonction `commande_t* nouvelle_commande(int nb_bouquets);` qui réserve la place en mémoire pour stocker un élément de type `commande_t` ainsi que pour stocker chacun des bouquets de la commande. La fonction retourne un pointeur qui cible cet espace mémoire. Attention, la structure contient un tableau et le tableau contient des pointeurs qui ciblent des bouquets. `nouvelle_commande` doit réserver de la mémoire pour tous les éléments mentionnés.

**Q3.** (1) Écrivez une fonction `void remplir_commande(commande_t* commande);` qui initialise la commande en mettant une fleur de chaque dans le bouquet `0`, deux fleurs de chaque dans le bouquet `1`, trois fleurs de chaque dans le bouquet `2` et ainsi de suite jusqu'à ce que tous les bouquets soient remplis.

**Q4.** (1) Écrivez une fonction `void affiche_commande(commande_t* commande);` qui affiche le contenu de la commande de la manière suivante :

```
La commande contient 3 bouquets.
Bouquet 1 :  1 rose, 1 tulipe et 1 pivoine
Bouquet 2 :  2 roses, 2 tulipes et 2 pivoines
Bouquet 3 :  3 roses, 3 tulipes et 3 pivoines
```

Note : On supposera que `commande` est une commande correctement remplie. Comme pour l'exercice précédent, on demande un respect du singulier et du pluriel lors de l'affichage.

**Q5.** (2) Écrivez une fonction `void libere_commande(bouquet_t* bouquet);` qui libère l'espace mémoire alloué pour la commande.

Attention : il faut bien libérer toute la mémoire que vous avez réservée !

**Q6.** (1) Écrivez une fonction `nb_de_fleurs` qui contiendra une commande en argument et permettra de récupérer au moyen d'autres arguments ou d'un retour de fonction le nombre total de chaque type de fleurs nécessaire à la commande.

Attention : Il y a donc trois informations à faire remonter en même temps. Dans l'exemple précédent, cela donnerait `6`, `6` et `6` mais les valeurs ne sont pas forcément identique à chaque fois.

**Q7.** (1) Écrivez une fonction `void exemple_de_commande(void);` dans laquelle vous définirez, remplirez et afficherez une commande de trois bouquets. Vous calculerez ensuite le nombre de fleurs de chaque type dont vous avez besoin grâce à `nb_de_fleurs` et afficherez ces nombres dans la fonction `exemple_de_commande`. Notez bien que la fonction `nb_de_fleurs` ne doit pas produire d'affichage, c'est bien la fonction `exemple_de_commande` qui contient les instructions d'affichage.