5 - Programmation en C

Si vous travaillez sur la machine virtuelle de l'IUT, il devient essentiel de sauvegarder vos travaux avant de partir, et même au milieu de la séance (pour parer aux pannes éventuelles...). À cet effet, vous pouvez utiliser Git (sans que cela ne soit obligatoire). Dans ce cas, créez un dépôt pour le module M2101 sur `git.unilim.fr` ou toute autre forge. Puis clonez-le ainsi :

```
git clone https://mon-super-dépôt.git
```

Les commandes à utiliser ensuite au fur et à mesure sont les suivantes :

```
git pull
git add *
git status
git commit -m 'message'
git push
```

1 Pointeurs

1.1 Un peu d'arithmétique

1. Dans un fichier **`M2101/Pointeurs/sum.c`**, écrivez deux implémentations de la fonction

   ```
   double sum (double *tab, int size)
   ```

   qui calcule la somme des éléments d'un tableau :
   a) une version `sum1` qui utilise un indice pour se déplacer dans le tableau ;
   b) une version `sum2` qui n'utilise pas d'indice auxiliaire (mais l'arithmétique des pointeurs).
2. Testez votre code. À cet effet, vous écrirez dans le même fichier une fonction

   ```
   int main (void)
   ```

   qui se chargera d'appeler vos fonctions **`sum1`** et **`sum2`** sur le tableau suivant :

```
[100, 10, 1, 0.1, 0.01]
```

1.2 Passage d'arguments

On considère le programme suivant :

```
#include <stdio.h>

void swap (double x, double y) {
    double z = x;
    x = y;
    y = z;
}

int main (void) {
    double a = 1;
    double b = 2;
    printf("a = %f et b = %f\n", a, b);
    swap(a, b);
    printf("a = %f et b = %f\n", a, b);
    return 0;
}
```

1. Essayez de deviner ce qui va s'afficher à l'exécution.
2. Récupérez le fichier **`M2101/Pointeurs/swap.c`** et vérifiez votre conjecture.
3. Écrivez une autre fonction **`swap2()`** qui réalise ce que son nom indique.

1.3 Allocation dynamique

1. Dans un fichier **`M2101/Pointeurs/array.c`** écrivez une fonction

```
int *create_array(int size)
```

qui crée un tableau contenant les entiers de $size$ à 1 (par ordre décroissant) et le retourne.

2. Dans le même fichier, écrivez une fonction

```
void display_array(int *array, int size)
```

qui affiche le contenu du tableau sous la forme suivante :

```
| 5 | 4 | 3 | 2 | 1 |
```

En fait, vous en écrirez deux versions, comme à la section 1.1 : l'une avec des indices, et l'autre avec l'arithmétique des pointeurs.

3. Dans ce même fichier, votre fonction **`main()`** devra :
   — appeler **`create_array()`** pour créer un tableau de taille 50 ;
   — appeler **`display_array1()`** et **`display_array2()`** pour l'afficher ;
   — libérer la mémoire allouée pour le tableau.

2 Compteurs

1. Dans un fichier **`M2101/Compteurs/counter.c`** écrivez une fonction

```
void print_hello()
```

qui affiche «Bonjour numéro i !» où i est un compteur qu'elle incrémente à chaque fois qu'elle est appelée.

2. Vous la testerez dans une fonction **`main`** qui l'appellera 10 fois. Notez que **`main`** ne passe aucune information à **`print_hello()`**.