Projet : Tris de Tableaux

Consignes :

- Le compte-rendu doit être rendu **sous forme numérique**.
- Le compte-rendu doit être soumis sur `Moodle` dans le cours `R1.01` puis `Projet` et évaluation.
- Le projet zippé contenant tous vos fichiers doit être rendu également.

Apprentissages critiques :	
AC0311	Utiliser un système informatique et ses outils
AC0312	Lire, exécuter, corriger et modifier un programme
AC0313	Traduire un algorithme, dans un langage et pour un environnement donné

L'objectif de ce projet est d'implémenter en langage C et en langage Python des tris de tableaux d'entiers, après avoir écrit leurs algorithmes.

N'oubliez pas de rendre votre travail après chaque séance sur Moodle !!!

Sommaire

1 Travail à Faire - Déroulement du Programme :

• Étape 1 :

Le programme demande à l'utilisateur un entier N strictement positif puis remplit un tableau de cette taille avec un nombre d'entiers aléatoires entre 0 et 10000.

Le programme vérifie la valeur entrée par l'utilisateur et réitère sa demande si l'utilisateur a fait une erreur. Un message d'erreur devra être de plus affiché.

• Étape 2 :

Les choix suivants sont proposés à l'utilisateur :
- Tri à bulles
- Tri par sélection
- Tri fusion
- Comparaison des tris

Le programme vérifie la valeur entrée par l'utilisateur et réitère sa demande si l'utilisateur a fait une erreur. Un message d'erreur devra être de plus affiché.

• Étape 3 :

Le programme demande à l'utilisateur si il souhaite trier son tableau par ordre croissant ou décroissant.

Le programme vérifie la valeur entrée par l'utilisateur et réitère sa demande si l'utilisateur a fait une erreur. Un message d'erreur devra être de plus affiché.

• Étape 4 :

L'algorithme de la procédure `tri_bulle` proposé en page 3 comporte des erreurs. Il faudra modifier les instructions de façon à ce que l'algorithme puisse répondre à l'objectif demandé.

Commentez dans votre programme C les différentes instructions des algorithmes des procédures `Fusion` et `TriFusion` en pages 3 et 4.

• Étape 5 :

Le programme affiche le tableau de départ puis le tableau trié. Le programme affiche aussi le nom du tri et le temps de traitement.

Une fois après avoir écrit votre algorithme, n'oubliez pas de le traduire en C, puis en Python.
Pour vérifier la validité de votre algorithme, il est conseillé de le traduire après chaque étape (hotmis l'étape 4).

2 Les Différents Tris :

• Le Tri par Sélection :

Le principe du **tri par sélection** d'un tableau de N éléments est le suivant :

Nous recherchons le maximum de tous les éléments (N), et nous le plaçons en dernière position en échangeant sa position avec l'élément placé en dernier.

Il ne reste plus qu'à trier les $N-1$ premiers éléments, pour lesquels nous répétons le même procédé.

• Le Tri à Bulles :

Principe :

Le **tri à bulles** ou tri par propagation est un algorithme de tri qui consiste à faire remonter progressivement les plus grands éléments d'un tableau, comme les bulles d'air remontent à la surface d'un liquide.

L'algorithme parcourt le tableau, et compare les couples d'éléments successifs.
Lorsque deux éléments successifs ne sont pas dans l'ordre croissant, ils sont échangés. Après chaque parcours complet du tableau, l'algorithme recommence l'opération. Lorsqu'aucun échange n'a lieu pendant un parcours, cela signifie que le tableau est trié. Nous arrêtons alors l'algorithme.

Algorithme proposé :

```
PROCÉDURE tri_bulle(tableau t)
    i ← longueur(t)
    echange ← faux
    TANT QUE ((i > 0) ET (echange=vrai)) FAIRE
        echange ← faux
        POUR j ALLANT DE 1 À i-1 PAS DE 1 FAIRE
            SI (t[j] < t[j + 1]) ALORS
                tmp ← t[j]
                t[j] ← t[j + 1]
                t[j] ← tmp
                echange ← faux
            FIN SI
        FIN POUR
        i ← i + 1
    FIN TANT QUE
FIN PROCÉDURE
```

• Le Tri Fusion :

Principe :

Ce tri est un autre exemple de méthode qui applique le principe "'diviser pour régner'". En effet, étant données deux suites d'éléments triés, de longueurs respectives $L1$ et $L2$, il est très facile d'obtenir une troisième suite d'éléments triés de longueur $L1 + L2$, par "interclassement" (ou fusion) des deux précédentes suites, comme illustré dans la procédure fusion.
Pour les besoins de la procédure `triFusion`, nous allons donner la forme suivante à la procédure `Fusion` qui interclasse deux suites d'éléments placés dans un tableau nommé *tab*, respectivement entre les indices *debut* et *mil* et entre les indices $mil + 1$ et n.

Algorithme de la procédure `Fusion` :

```
PROCÉDURE Fusion(entier[] tab, entier[] tmp, entier debut, entier mil, entier fin)
    entier k
    entier i ← debut
    entier j ← mil + 1
    POUR k ALLANT DE debut À fin PAS DE 1 FAIRE
        SI ((j > fin) OU ((i ≤ mil) ET (tab[i] < tab[j]))) ALORS
            tmp[k] ← tab[i]
            i ← i + 1
        SINON
            tmp[k] ← tab[j]
            j ← j + 1
        FIN SI
    FIN POUR
    POUR k ALLANT DE debut À fin PAS 1 FAIRE
        tab[k] ← tmp[k]
    FIN POUR
FIN PROCÉDURE
```

Algorithme de la procédure `TriFusion` :

```
PROCÉDURE triFusionI(entier[] tab)
    entier[] tmp ← tableau de taille N
    entier i ← 1
    entier debut ← 1
    entier fin ← debut + i + i − 1
    TANT QUE (i < N) FAIRE
        debut ← 1
        TANT QUE (debut + i − 1 < N) FAIRE
            fin ← debut + i + i − 1
            SI (fin > N) ALORS
                fin ← N
            FIN SI
            Fusion(tab, tmp, debut, debut + i − 1, fin)
            debut ← debut + i + i
        FIN TANT QUE
        i ← i + i
    FIN TANT QUE
FIN PROCÉDURE
```

• La Fonction de Comparaison :

Cette option proposée à l'utilisateur permet de comparer les tris.

Si l'utilisateur choisit l'option comparaison de tris, le programme trie le tableau de départ avec chaque tri et stocke la durée de chacun d'entre eux.

Ensuite le programme affiche à l'utilisateur la liste des tris du plus performant au moins performant en indiquant leur nom et leur durée.

3 Améliorations du Programme :

- **Nombres Aléatoires :**

Modifier votre programme afin qu'il demande à l'utilisateur de délimiter la plage de nombres aléatoires.

- **Chaînes de Caractères :**

Modifier votre programme pour que l'utilisateur utilise des chaines de caractères afin de choisir le type de tri ou l'ordre (croissant ou décroissant).

L'utilisateur devra utiliser un mot.

- **Exportation dans un Fichier :**

Il sera possible de conserver dans un fichier texte les données du tableau avant le tri, et dans un autre fichier les données du tableau après le tri.

- **Tableau $N \times N$:**

Le tableau n'est plus seulement à une dimension mais de dimension $N \times N$.

L'algorithme permettra de trier par ordre croissant ou décroissant les contenus des lignes ou des colonnes de ce tableau suivant le choix de l'utilisateur.

4 Annexes :

• Déclaration du tableau :

Pour déclarer votre tableau en allouant dynamiquement de la mémoire en langage C, vous devez utiliser les instructions suivantes :

```
int *tableau;
tableau = (int(*))malloc(N*sizeof(int));
```

Cette instruction alloue dynamiquement de la mémoire à votre tableau d'entiers de taille N.
Il faut ajouter la librairie `stdlib.h`.

• Valeurs Aléatoires :

Pour les valeurs aléatoires en langage C, vous devez utiliser la librairie `time.h`.

Un exemple d'utilisation est :

```
srand(time(NULL));
tableau[i]=rand();
```

• Variables Temporelles :

- Langage C :

En utilisant la bibliothèque `time.h`, le type `clock_t` représente le nombre de "tick" écoulé.
Le tick est une unité de mesure propre au système considéré.
Il est possible de connaître le nombre de ticks par seconde en interrogeant la constante `CLOCKS_PER_SEC`.

Si nous souhaitons de plus connaître le nombre de ticks consommé par votre programme, il faut utiliser la fonction `clock()`.

- Langage Python :

Il est conseillé d'utiliser la fonction `time` en important le module `time`.