TP5

1. Complétez le squelette de l'algorithme **`Fusion_tab_trie`** (au verso). L'action réalise la fusion de deux tableaux de réels triés nommés reelTab1 et reelTab2, en un tableau trié (nommé reelTabF).
2. Ecrivez, en C++, la procédure **`Fusion_tab_trie`** qui réalise la fusion de deux tableaux de réels triés, en un tableau trié.
```
void fusion_tab_trie (float tab1[], int taille1, float tab2[],
            int taille2, float tab_fusion[], int &taille_fusion)
```
3. Ecrivez, en C++, une fonction **`Fusion`** qui réalise la fusion uniquement si (1) les deux tableaux en paramètre sont triés dans l'ordre croissant et (2) si le tableau obtenu par fusion aura une taille inférieure à `TailleMax`. Si la fusion est effectuée la fonction retourne `true` sinon elle retourne `false`.
```
bool fusion (float tab1[], int taille1, float tab2[], int taille2,
      float tab_fusion[], int &taille_fusion)
```
4. Testez votre fonction `fusion` avec les tableaux suivants. Pour cela, utilisez la fonction `Lecture_Tableau` et la procédure `Afficher_Tableau` qui sont dans le fichier `/net/Bibliotheque/AS/TP_ASD_PROG/tp2.cc`.
5. Expliquez pour chaque test son intérêt (quelle erreur permet-il de détecter ?).

   T1 = [1, 3, 5, 11, 9] et T2 = [0, 2, 4, 6, 8, 10]
   T1 = [1, 3, 5, 7, 9, 11, 13] et T2 = [0, 2, 5, 6, 8, 10, 12]
   T1 = [1, 3, 5, 7, 9, 11, 13] et T2 = [0, 2, 6, 8, 10, 14, 12]
   T1 = [1, 2, 3, 4, 5, 6, 7] et T2 = [14, 15, 16, 17, 18, 19]
   T1 = [14, 15, 16, 17, 18, 19] et T2 = [1, 2, 3, 4, 5, 6, 7]
   T1 = [0, 2, 4, 6, 8, 10, 12] et T2 = [1, 3, 5, 7, 9, 11, 13]
   T1 = [0.01, 0.02, 0.21, 2.16, 21.68] et T2 = [0.04, 2,09, 2.13, 21.61, 216.85]

L'algorithme vérifiant qu'un tableau de réels est effectivement trié est présenté ci-dessous.

```
Fonction Verifier_Tri ( E reelTab : tableau de réels, E tailleTab entier)
      retourne un booléen
// Vérifie si un fichier de réels est trié dans l'ordre croissant ou non
variables
       courant, pred : réels
       trié : booléen
       index : entier
début
  index ← 1
  trié ← Vrai
  si index <= talleTab Alors
      debut
            pred ←  reelTab[index]
            index ← index + 1
       fin
  Tant Que index <= tailleTab et trié
   Faire début
       courant ←  reelTab[index] ; index ← index + 1
 Si courant < pred Alors trié ← Faux
       Sinon  pred ← courant
   fin
   retourner trié
fin
```

```
Action Fusion_tab_trie (E reelTab1 : tableau de réels, E tailleTab1 entier,
                        E reelTab2 : tableau de réels, E tailleTab2 entier,
                        S reelTabF : tableau de réels, E tailleTabF entier )

variables
      index1, index2 indexF : entiers ;
début
      tailleTabF = tailleTab1+tailleTab2 ;

      index1  ← 1 ; index2 ← 1 ; indexF ← 1 ;
      Tant Que index1 <= tailleTab1  et index2 <= tailleTab2
      Faire début
            si reelTab1[index1] < reelTab2[index2]
            Alors début









                  fin
            Sinon début









                  fin
      fin
      //On est arrivé à la fin d'au moins l'un des deux tableaux
      Tant Que index1 <= tailleTab1
      Faire début









      fin
      Tant Que index2 <= tailleTab2
      Faire début









      fin
fin
```