# Travaux Dirigés n°4

Pour chaque problème, il vous est demandé de définir clairement :

- les données d'entrée du problème en précisant leurs types (nombre entier, réel, ...) ;
- les éventuelles données de sortie du problème en précisant leurs types ;
- les instructions permettant d'obtenir les données de sortie à partir des données d'entrée.

Tester ensuite votre algorithme à la main à partir de données d'entrées judicieusement choisies pour explorer les différents cas de fonctionnement.
L'objectif des exercices de ce TD est de trier un tableau d'entiers $arr$ de $N$ éléments.

**Exercice 1**

**Tri par sélection**
Écrire l'algorithme de tri par sélection qui se base sur le principe suivant. On sélectionne tout d'abord l'élément le plus petit du tableau, c.à.d. on trouve l'entier $p$ tel que $\forall 1 \leq i \leq N, arr[i] \geq arr[p]$. Une fois cet emplacement trouvé, on échange les élements $arr[1]$ et $arr[p]$. Puis on recommence ces opérations pour le reste du tableau (c.à.d. les éléments compris entre les indices 2 et $N$. On recherche alors le plus petit élément de cette nouvelle suite de nombre et on échange avec $arr[2]$. Et ainsi de suite ... jusqu'au moment où on a placé tous les élements du tableau.
Exemple d'exécution :
Tableau initial : 5 3 2 4 8.
Les itérations

2 3 5 4 8
2 3 5 4 8
2 3 4 5 8
2 3 4 5 8

Solution

Entrée : *arr* : tableau d'entiers ; $N$ : le nombre d'éléments de *arr*
Sortie : *arr* triés.

```
int i,j,minIndex,tmp;
for (i = 0; i < N - 1; i++) {
            minIndex = i;
            for (j = i + 1; j < N; j++)
                  if (arr[j] < arr[minIndex])
                        minIndex = j;
            if (minIndex != i) {
                  tmp = arr[i];
                  arr[i] = arr[minIndex];
                  arr[minIndex] = tmp;
            }
}
```

**Exercice 2**

**Tri à bulle**
Le principe est de comparer successivement tous les éléments adjacents d'un tableau et de les échanger si le premier élément est supérieur au second. On recommence cette opération tant que tous les éléments ne sont pas triés.
Exemple d'exécution :
Tableau initial : 5 3 2 4 8.
Les itérations

3 2 4 5 8
2 3 4 5 8
2 3 4 5 8

Solution

Entrée : *arr* : tableau d'entiers ; $N$ : le nombre d'éléments de *arr*
Sortie : *arr* triés.

```
int i,j,echange,tmp;
echange=1;
j=0;
while (echange) {
            echange = 0;
            j++;
            for ( i = 0; i < N - j; i++) {
                  if (arr[i] > arr[i + 1]) {
                        tmp = arr[i];
                        arr[i] = arr[i + 1];
                        arr[i + 1] = tmp;
                        echange = 1;
                  }
            }
 }
```

**Exercice 3**

**Tri par insertion**
Le tri par insertion consiste à prendre l'élément se trouvant juste après la partie déjà triée du tableau et de trouver sa place dans cette dernière. Le premier élément à trier est le deuxième élément du tableau, le premier étant forcément déjà trié puisqu'il est tout seul. On recommence ce procédé jusqu'au dernier élément du tableau.
Exemple d'exécution :
Tableau initial : 5 3 2 4 8.
Les itérations

3 5 2 4 8
2 3 5 4 8
2 3 4 5 8
2 3 4 5 8

Solution

Entrée : *arr* : tableau d'entiers ; $N$ : le nombre d'éléments de *arr*
Sortie : *arr* triés.

```
int i,j,tmp;
for (i = 1; i < N; i++) {
    j = i;
    while (j > 0 && arr[j - 1] > arr[j]){
          tmp = arr[j];
          arr[j] = arr[j - 1];
          arr[j - 1] = tmp;
          j--;
    }
}
```

**Exercice 4**

**Le tri par paquets**

Cet algorithme de tri ne s'applique qu'aux entiers dont la valeur est comprise entre 0 et une constante $R\_MAX$. On utilise en outre du tableau de départ, un autre tableau $P$ de $R\_MAX+1$ éléments initialisés à 0. Les éléments de $P$ sont appelés les paquets. La première étape de l'algorithme consiste à déterminer le plus petit élément *mini* et le plus grand élément *maxi*. La deuxième étape calcule pour tout $mini \leq i \leq maxi$ la valeur du paquet $P[i]$ qui correspond au nombre d'occurence de la valeur $i$ dans le tableau de départ. Enfin, l'algorithme reécrit entièrement le tableau à trier de départ à l'aide des paquets dans $P$. En gros, on prend le premier paquet $P[mini]$, et on écrit la valeur *min*, $P[mini]$ fois. On recommence avec le dexuième paquets $P[mini+1]$, ...

Solution

Entrée : $arr$ : tableau d'entiers ; $N$ : le nombre d'éléments de $arr$ ; $R\_MAX$ la valeur maximale
Sortie : $arr$ triés.

```
int P[R_MAX+1];
int i,j,k,mini,maxi;
mini=arr[0];
maxi=arr[0];
for (i=1;i<N;i++){
    if (arr[i]>maxi)
       maxi=arr[i];
    if (arr[i]<mini)
       mini=arr[i];
}
for (i=mini;i<=maxi;i++)
    P[i]=0;
for (i=0;i<N;i++)
    P[arr[i]]++;
i=0;
for (j=mini;j<=maxi;j++){
    for (k=0;k<P[j];k++){
        arr[i]=j;
        i++;
    }
}
```