Langage C

Sujet 00a : Algorithmes de tri de tableaux

L'objectif de ce TD est d'étudier diverses méthodes permettant de trier, par ordre croissant, les valeurs d'un tableau.

1 Méthode de tri par recherches successives de minimum (ou par sélection)

Le principe de cette méthode très intuitive consiste à :
- chercher le minimum dans un sous-tableau (au départ le tableau complet contenant les N valeurs non ordonnées)
- permuter ce minimum avec le premier élément du sous-tableau
- puis itérer ce traitement sur un nouveau sous-tableau de N-1 éléments (on ne tient plus compte du premier élément qui est maintenant à sa place)

1. Ecrire la fonction `Permut(A,B)` qui permute les valeurs réelles contenues dans `A` et `B`.
2. En s'inspirant de la fonction `MinTab(N,Tab)` écrite dans le TD précédent, écrire maintenant la fonction `RangMinTab(N,P,Tab)` qui retourne cette fois **le rang** (et non la valeur) de la plus petite des valeurs réelles contenues dans le tableau `Tab` **à partir du rang** `P`.
3. Ecrire la fonction `TriSel(N,Tab)` qui permet de trier par ordre croissant, suivant la méthode énoncée, les `N` valeurs réelles contenues dans le tableau `Tab`.
4. Eléments de complexité algorithmique : évaluer le nombre d'opérations réalisées par cette fonction (en fonction de `N`).

2 Méthode de tri par bulles (ou par propagation)

Le principe de cette méthode consiste à comparer les couples de valeurs successives `Tab[i]` et `Tab[i+1]` pour `i` variant de `0` à `N-2`, et à les permuter si elles sont mal ordonnées. L'algorithme s'arrête lorsque l'on constate qu'aucune permutation n'a été effectuée lors du dernier "survol" du tableau.

1. Ecrire la fonction `TriBul(N,Tab)` qui permet de trier, suivant la méthode énoncée, les `N` valeurs réelles contenues dans le tableau `Tab`. La fonction `Permut(A,B)` écrite précédemment pourra bien évidemment être réutilisée.
2. Eléments de complexité algorithmique : évaluer le nombre d'opérations réalisées par cette fonction, et comparer ce résultat avec celui de la première méthode de tri. Conclusion ?

Solution

Pour cet exercice nous réutilisons les fonctions `ChargerTab` et `EditerTab` du sujet 00.

1. Tri par sélection

```
#include <stdio.h>

void Permut(float *A, float *B) {
  float temp;

  temp = *A;
  *A = *B;
  *B = temp;
}

int RangMinTab(int N, int P, float Tab[]) {
  int i,resultat=P;

  for (i=P+1;i<N;i++)
    if (Tab[i]<Tab[resultat])
      resultat=i;

  return resultat;
}

void TriSel(int N, float Tab[]) {
  int i;

  for (i=0;i<N-1;i++) {
    int min = RangMinTab(N,i,Tab);
    Permut(&Tab[i], &Tab[min]);
  }
}

void ChargerTab(int* N, float Tab[]) {
  int i;

  printf("Combien de valeurs ? ");
  scanf("%d",N);

  for(i=0;i<*N;i++) {
    printf("Valeur %d: ",i+1);
    scanf("%f",&Tab[i]);
  }
  printf("\n");
}
```

```
void EditerTab(int N, float Tab[]) {
  int i;
  for (i=0;i<N;i++) {
    printf("Valeur %d = %f\n",i+1,Tab[i]);
  }
  printf("\n");
}

int main(int argc, char** argv) {
  float myTab[100];
  int   nb;

  ChargerTab(&nb,myTab);
  EditerTab(nb,myTab);
  TriSel(nb,myTab);
  EditerTab(nb,myTab);
}
```

2. Tri à bulles

```
#include <stdio.h>

void Permut(float *A, float *B) {
  float temp;

  temp = *A;
  *A = *B;
  *B = temp;
}

void TriBul(int N, float Tab[]) {
  int i,nbp;

  do {
    nbp=0;

    for (i=0;i<N-1;i++)
      if (Tab[i+1]<Tab[i]) {
        Permut(&Tab[i], &Tab[i+1]);
        nbp++;
      }
  } while (nbp!=0);
}
```

```
void ChargerTab(int* N, float Tab[]) {
  int i;

  printf("Combien de valeurs ? ");
  scanf("%d",N);

  for(i=0;i<*N;i++) {
    printf("Valeur %d: ",i+1);
    scanf("%f",&Tab[i]);
  }
  printf("\n");
}

void EditerTab(int N, float Tab[]) {
  int i;
  for (i=0;i<N;i++) {
    printf("Valeur %d = %f\n",i+1,Tab[i]);
  }
  printf("\n");
}

int main(int argc, char** argv) {
  float myTab[100];
  int   nb;

  ChargerTab(&nb,myTab);
  EditerTab(nb,myTab);
  TriBul(nb,myTab);
  EditerTab(nb,myTab);
}
```