TD 5 : Les tris

1. Le plus petit

On considère le tableau T de n entiers. Chercher le rang et la valeur du plus petit élément.

2. Tri par permutation

Soit n la dimension d'un tableau d'entiers T déjà initialisé. On suppose que T est partiellement trié et que les propriétés suivantes sont vraies pour un indice k dans [0, n-1[:

- T est trié (ordre croissant) de 0 à l'indice k non compris,
- quels que soient i dans [0, k[et j dans [k, n[, $T[i] \leq T[j]$

Exemple d'une suite de valeurs partiellement triée : 3, 10, 16, 18, *48, 19, 21, 37, 27*

Ces deux propriétés peuvent être étendues à k+1 en appliquant le traitement suivant :

- chercher la position jMin du plus petit élément entre les indices k et n (non compris n…),
- permuter le contenu des cases k et jMin.

Mettre en œuvre la méthode de tri utilisant ce principe (on remarquera que ces propriétés sont vraies pour k=0 et que le tableau est totalement trié si k=n-1)

3. Recherche d'un élément dans un tableau ordonné

On suppose qu'un tableau T contient N entiers triés dans l'ordre croissant de 0 à N-1.
Ecrire la suite d'instructions qui permet de trouver par dichotomie (cf exo 1, TD 2) la position (si elle existe) d'un entier v dans T.
Dans la conception de l'algorithme, on définira deux bornes Min et Max qui délimite la zone de recherche et on fera en sorte de respecter la propriété ci-dessous à chaque itération : *Si v est présent, alors sa position i dans T est telle que* $Min \leq i < Max$

4. Fusion *(si temps disponible)*

Ecrire le programme qui étant donnés deux tableaux triés T1[N] et T2[M] , construit un tableau trié T3[M+N] en fusionnant T1 et T2.

Correction TD 5 : Les tris

Le plus petit

Voir tri par permutation.

Tri par permutation

Algorithme à affiner :

- rechercher le minimum de la liste (tableau)
- le permuter avec le premier élément (rang 0) de la liste
- recommencer sur la liste restante jusqu'à épuisement

```
#include <stdio.h>
#define MAX 100
int main(void)
{ int i,j,jMin, n, tmp ;
  int t[MAX];
  printf("donnez une suite d'entiers positifs terminee par 0\n") ;
  n=0;
  do
    scanf("%d", &t[n]) ;
  while ((t[n] != 0) && (++n < MAX)) ;

  for (i=0 ; i<n-1 ; i++)
  { /* recherche du plus petit au dela de i */
    jMin = i;
    for (j=i+1 ; j<n ; j++)
      if (t[j] < t[jMin]) jMin = j;
    /* permutation */
    tmp = t[i];
    t[i] = t[jMin];
    t[jMin] = tmp;
  }

  printf("tableau trie :\n");
  for (i=0 ; i<n ; i++)
    printf("%d\n", t[i]);
  return 0 ;
}
```

Recherche d'un élément dans un tableau ordonné

On suppose qu'un tableau T contient N entiers triés dans l'ordre croissant de 0 à N-1.
Ecrire la suite d'instructions qui permet de trouver par dichotomie (cf exo 1, TD 2) la position (si elle existe) d'un entier v dans T .
Dans la conception de l'algorithme, on définira deux bornes Min et Max qui délimite la zone de recherche et on fera en sorte de respecter la propriété ci-dessous à chaque itération : *Si v est présent, alors sa position i dans T est telle que* $Min \leq i < Max$

Algorithme à affiner :

- comparer la valeur du milieu avec la valeur recherchée
- réduire l'intervalle [Min, Max[des solutions à la première moitié ou la deuxième moitié en fonction du résultat de la comparaison
- recommencer jusqu'à obtenir un intervalle qui ne contient qu'une valeur ([Min, Min+1[ )

```
#include <stdio.h>
#define N 20
int main(void)
{ int min, max, milieu, cpt, val ;
  int t[N]={1,2,4,5,8,9,25,30,41,52,53,54,55,56,60,61,70,72,74,76} ;

  printf("quelle valeur recherchez-vous\n");
  scanf("%d", &val);

  /* cas particulier */
  if (val<t[0])
  { printf("element absent (a inserer en t[0])\n") ;
    return 0 ;
  }

  /* cas général : la solution se trouve dans [min, max[ */
  cpt=0;
  min = 0;
  max = N;
  while (max-min > 1)
  { milieu = (min+max)/2;
    if (val<t[milieu]) max = milieu;
    else min = milieu;
    cpt++;
  }
  if (t[min]==val)
    printf("j'ai trouve t[%d]=%d en %d coups\n", min, val, cpt);
  else
    printf("element absent (a inserer en t[%d])\n", max);
  return 0 ;
}
```

Fusion

Ecrire le programme qui étant donnés deux tableaux triés T1[N] et T2[M] , construit un tableau trié T3[M+N] en fusionnant T1 et T2.

- Prendre le 1er élément de T1 et le 1er de T2
- jusqu'à vider une des 2 listes
  - mettre le plus petit des deux dans T3
  - prendre le suivant du plus petit dans la liste correspondante
- recopier la fin de la liste T1 dans T3
- recopier la fin de la liste T2 dans T3

```c
#include <stdio.h>
#define N1 5
#define N2 4

int main(void)
{ int i1,i2 ; // indices de l'élément courant de t1 et t2
  int i3 ; // nbre d'éléments déjà recopiés dans t3
           // c'est aussi la 1ère place dispo pour un nouvel élém.
  int t1[N1]={1,10,22,52,53} ;
  int t2[N2]={8,9,25,60} ;
  int t3[N1+N2];
  i1=i2=i3=0;
  /* fusion */
  while (i1<N1 && i2<N2)
    if (t1[i1]<t2[i2]) /* style "Pascal" :*/
    { t3[i3] = t1[i1];
      i1 = i1+1;
      i3 = i3+1;
    }
    else /* style "C" : */
      t3[i3++] = t2[i2++];
  // fin while
  /* vider le tableau restant */
  while (i1<N1) t3[i3++] = t1[i1++];
  while (i2<N2) t3[i3++] = t2[i2++];

  for (i3=0 ; i3< N1+N2 ; i3++)
    printf("%d ", t3[i3]);
  printf("\n");
  return 0 ;
}
```