COMPTE RENDU EXO 2 TP 2 RN40

BARTHELME ALEXANDRE

Mon code complet est disponible sur mon GitHub à l'adresse :
https://github.com/Iron68A/RN40/tree/main/TP/TP2

Ou par capture d'écran

I) La recherche séquentielle

Elle est beaucoup plus simple et moins longue à mettre en place (voir capture d'écran). Cette fonction de recherche compare chaque élément du tableau avec la valeur donnée par l'utilisateur. Si la valeur est égale, le programme affiche la position, mais ne s'arrête pas dans le cas où la même valeur reviendra plus tard dans le tableau

II) La recherche dichotomique

Cette fonction va d'abord trier dans un ordre croissant le tableau (en utilisant une variable temporaire), puis va chercher le milieu du tableau. Après, elle regarde si la valeur est plus grande ou non que la valeur à l'indice « milieu » du tableau. En fonction de ça, la fonction fera une recherche séquentielle sur la partie supérieure ou inférieur du tableau. L'avantage de cette méthode est que le traitement sera plus court, même s'il doit d'abord trier le tableau, ce programme est plus « léger » car on coupe le tableau en deux. Sur de très grande opération, il est préférable d'utiliser cette méthode que de comparer chaque élément avec le suivant. Si on a un tableau de 100 valeurs, il faut toutes les comparer. Avec l'autre technique, on ne fait que 50 comparaisons.

III) Captures d'écran :

Recherche séquentielle :

```c
void recherchetab(float tableau[], int taille, float valeur)
{
    for (int i = 0; i < taille; i++)
    {
        if (tableau[i] == valeur)
        {
            printf("La valeur %f est dans le tableau à la position %d \n", valeur, i);
            printf("PS : Les indices commencent a 0 \n");
        }
    }
}
```

Recherche dichotomique :

```c
void dichotab(float tableau[],int taille, float valeur){
//trier le tableau
    for (int i = 0; i < taille; i++)
    {
        for (int j = 0; j < taille; j++)
        {
            if (tableau[i] < tableau[j])
            {
                float temp = tableau[i];
                tableau[i] = tableau[j];
                tableau[j] = temp;
            }
        }
    }
    printf("Le tableau trie est: ");
    for (int i = 0; i < taille; i++)
    {
        printf("%f ", tableau[i]);
    }
```

```c
43  //trouver milieu puis valeur
44      int milieu = taille/2;
45      printf("miliue = %d et contenu = %f \n", milieu, tableau[milieu]);
46      if (tableau[milieu] == valeur)
47      {
48          printf("La valeur %f est dans le tableau à la position %d \n", valeur, milieu);
49
50      }
51      else if (tableau[milieu] > valeur)
52      {
53          for (int i = 0; i < milieu; i++)
54          {
55              if (tableau[i] == valeur)
56              {
57                  printf("La valeur %f est dans le tableau à la position %d \n", valeur, i);
58
59              }
60          }
61      }
62      else if (tableau[milieu] < valeur)
63      {
64          for (int i = milieu; i < taille; i++)
65          {
66              if (tableau[i] == valeur)
67              {
68                  printf("La valeur %f est dans le tableau a la position %d \n", valeur, i);
69
70              }
71          }
72      }
73      else printf("La valeur %f n'est pas dans le tableau \n", valeur);
```

Le main :

```c
//Main
int main(){
    int i =0;
    printf("Entrez la taille du tableau puis son contenu :");
    scanf("%d", &i);
    float tableau[i];
    for (int j = 0; j < i; j++)
    {
        printf("Entrez la valeur %d: ", j);
        scanf("%f", &tableau[j]);
    }
    float valeur;
    printf("Entrez maintenant la valeur a rechercher : ");
    scanf("%f", &valeur);
    printf("valeur chercher par methode de comparaison : \n");
   recherchetab(tableau, i, valeur);
    printf("valeur chercher par methode de dichotomie : \n");
    dichotab(tableau, i, valeur);
    return 0;

}
```

Pour toutes questions ou informations complémentaires, me contacter par mail :

alexandre.barthelme@utbm.fr