
TD 5

Exercice n° 2

```
#include <stdio.h>
#include <stdlib.h>

/* affiche les n premières valeurs d'un tableau */
void affiche(int t[], int n)
{
    int i;

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

main()
{
    int taille1, taille2;
    int *t1, *t2;

    printf("\nEnter une valeur pour le tableau 1");
    scanf("%d",&taille1);
    printf("\nEnter une valeur pour le tableau 2");
    scanf("%d",&taille2);

    /* allocation mémoire */
    t1 = (int *) malloc (taille1*sizeof(int));
    t2 = (int *) malloc (taille2*sizeof(int));

    int i;

    /* mettre des valeurs aléatoires dans les tableaux */
    for(i = 0; i < taille1; i++)
        t1[i] = rand() % 100;
    for(i = 0; i < taille2; i++)
```

```

        t2[i] = rand() % 100;

    affiche(t1, taille1);
    affiche(t2, taille2);

    /* fusion des deux tableaux */
    int * t3, *debut;

    t3 = (int *) malloc ((taille1+taille2)*sizeof(int));

    debut = t3;
    for(i = 0; i < taille1 + taille2; i++)
        if (i < taille1)
            *t3++ = *t1++;
        else
            *t3++ = *t2++;

    t1 = debut;

    affiche(t1, taille1 + taille2);
}

```

Exercice n° 3

```

#include <stdio.h>
#include <stdlib.h>

#define N_MAX 50

/* affiche les n premières valeurs d'un tableau */
void affiche(int t[], int n)
{
    int i;

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

main()
{
    int x, i;
    int *t1, *t2, *t;

```

```

int * p_tableau;

t1 = (int *) malloc(N_MAX*sizeof(int));
t2 = (int *) malloc (N_MAX*sizeof(int));

t = t1;
/* mettre des valeurs aléatoires dans le tableau */
for(i = 0; i < N_MAX; i++)
    *t++ = rand() % 3;

affiche(t1, N_MAX);

printf("\nVeuillez entrer un entier X");
scanf("%d",&x);

p_tableau = t2;
t = t1;

for(i = 0; i < N_MAX; i++)
    if ( (*t1) != x)
        *t2++ = *t1++;
    else
        t1++;

t2 = p_tableau;

affiche(t2, N_MAX);
}

```

Exercice n° 4

```

#include <stdio.h>
#define MAX 100

int main(int argc, char *argv[]) {

    void merge(int a[], int b[], int c[], int m, int n);
    void affiche(int *t, int n);
    void mergesort(int t[], int n);

```

```

void usage(char *s);

int b[] = {4,3,1,67,55,8,0,4,-5,37,7,4,2,9,1,-1};
int r[MAX];
int m = 16;

affiche(b,16);
mergesort(b,16);
affiche(b,16);
}

void merge(int a[], int b[], int c[], int m, int n){
    int i,j,k; /* les compteurs */

    i = j = k = 0;

    while (k < m+n){

        /* cas où a est vide */
        if (i >= m)
            c[k++] = b[j++];

        /* cas où b est vide */
        if (j >= n)
            c[k++] = a[i++];

        if (i < m && j < n){
            if (a[i] < b[j])
                c[k++] = a[i++];
            else
                c[k++] = b[j++];
        }
    }
}

void mergesort(int t[], int n){
    int i = 1, j;
    int t_temp[MAX]; /* tableau temporaire */
    int *p_tableau1, *p_tableau2; /* pointeurs de tableau */

    /* vérification que la taille du tableau est un multiple de 2*/
    while (i < n)
        i *= 2;

    for (i = 1; i <= n/2; i *= 2){
        /* fusion du tableau, il faut couper au bon endroit */
        for (j = 0; j < n ; j += (2*i))

```

```

merge(&t[j],&t[j+i],&t_temp[j],i,i);

p_tableau1 = t;
p_tableau2 = t_temp;
/* copie du tableau temporaire */
for (j = 0; j < n; j++)
    /* t[j] = t_temp[j]; */
    *(p_tableau1++) = *(p_tableau2++);
}
}

```

```

void affiche(int *t, int n){
    int i;

    for (i = 0; i < n; i++)
        printf("%d\t",*(t++));
    printf("\n");
}

```