Exercice 1 sur les pointeurs et fonctions

Écrire une fonction qui permet de rechercher dans un tableau d'entiers tab une valeur A. void chercherVal (int tab[], int n, int A, int *pos, int *nb_occ); Dans pos, la fonction sauvegarde l'indice de la dernière apparition et -1 si la valeur n'a pas été trouvée. Dans nb_occ, elle sauvegarde le nombre d'occurence de A dans tab.

Exercice 2 sur les pointeurs et fonctions

On souhaite écrire une fonction qui permet de résoudre une équation du premier degré : ax+b=0. Voici le prototype de la fonction : int resoudre1(int a, int b, float *x);

la fonction retourne le nombre de solution trouvé (0: pas de solution, 1: une solution, -1: tout x est solution). Dans le cas où l'équation a une solution, la fonction retourne la solution dans x.

Exercice 3

1. Écrire une fonction : supprimer_nul, qui permet de supprimer la première valeur nulle d'un tableau d'entier passé en paramètre.

2. Écrire une fonction : nb_occurrence, qui étant donnés un tableau T de n entiers et un entier x, détermine puis retourne le nombre d'occurence de x dans T.

3. Écrire une fonction : compacter, qui permet de compacter les éléments du tableau tab. Cette opération consiste à supprimer toutes les valeurs nulles du tableau.

Astuce: utiliser nb_occurrence pour trouver nb, nombre de répétition de zéro dans le tableau, puis appeler supprimer_nul nb fois.

Exercice 4

Écrire une fonction qui détermine les indices de la plus grande valeur dans imax et la plus petite valeur dans imin d'un tableau d'entiers.

void maxima (int tab[], int n, int *imax, int * imin);

Si le tableau contient plusieurs maxima ou minima, la fonction retiendra la position du premier maximum ou minimum rencontré.

Exercice 5

Écrire une fonction qui détermine si une matrice carrée est symétrique ou non. La fonction retourne 1 si oui et 0 si non.

Une matrice est symétrique si Mij = Mji pour 0<=i<n et 0<=j<n.

Voici le prototype de la fonction: int estSymetrique (int M[MAX][MAX], int n);

Avec MAX est une constante.

Exercice 6

Écrire une fonction qui permet de rechercher dans un tableau d'entiers tab une valeur A. int chercherTab (int tab[], int n, int A, int *pos);

La fonction retourne le nombre d'occurrence si A existe dans le tableau et 0 si non. Dans pos, elle retourne l'indice de la dernière apparition de la valeur dans le tableau et -1 si la valeur n'a pas été trouvée.

Exercice 7

Soit une matrice A à deux dimensions NxN. Un « point col » est un élément de la matrice qui est minimum de sa ligne et maximum de sa colonne ou inversement.

1. Ecrire une fonction estMaxLigne qui retourne 1 si une valeur M est la plus grande sur toute la ligne L.

2. Ecrire une fonction estMinColonne qui retourne 1 si une valeur M est la plus petite sur toute la colonne C.

3. Ecrire une fonction chercherPointCol qui affiche les coordonnées de tous les points cols d'une matrice A. La fonction retourne le nombre de point col trouver.

Voici les prototypes des fonctions demandées :

```
int estMaxLigne (int A[][], int N, int M, int L);

int estMinColonne (int A[][], int N, int M, int C);

int chercherPointCol (int A[][], int N);
```

Exercice 8

On souhaite écrire une fonction qui permet de résoudre une équation du second degré. Voici le prototype de la fonction:

```
int resoudre2(int a, int b, int c, float *x1, float *x2);
```

la fonction retourne le nombre de solution trouvé (0: pas de solution, 1: une solution, 2: une solutions, -1: tout x est solution). Dans le cas où l'équation a une solution, la fonction retourne la solution dans x1. Dans le cas où l'équation a deux solutions, la fonction retourne les solutions dans x1 et x2.

Solution de l'exercice 1

```
#include<stdio.h>
#define N 100
void chercherVal (int tab[], int n, int A, int *pos, int *nb_occ);
void saisie(int *ptab,int *pn); //ou int tab[]
void affiche(int tab[],int n);
void main()
```

```
{
int tab[N],n,val,nb_occ=0,pos=-1;
saisie(tab,&n);
affiche(tab,n);
printf("\n Donner la val a chercher \n"); // SERT LORSKE unr fction A appelle B et B appelle A
scanf("%d",&val);
chercherVal(tab,n,val,&pos,&nb_occ);
if(nb_occ==0 ) printf("NEXISTE PAS");
else printf("Existe %d fois à la position %d ",nb_occ,pos);
}
void chercherVal (int tab[], int n, int A, int *pos, int *nb_occ)
{
  int i=0;
while(i<n)
   {
      if(tab[i]==A)
      {
  //nb_occ=*nb_occ+1;
        (*nb_occ)++;
         *pos=i;
      }
 i++;
   }
}
void saisie(int *ptab,int *pn) //ou int tab[]
{
   int i;
   do{
      printf("Donner la taille du tableau \n");
```

```
 scanf(“%d”,pn);
    } while(*pn<=0);
    for(i=0;i<*pn;i++)
    {
  printf(“Donner tab[%d] “,i);
        scanf(“%d”,ptab+i); // OU &tab[i]
    }
}
void affiche(int tab[],int n)
{
     int i;
     for(i=0;i<n;i++)
    {
        printf(” %d | “,tab[i]);
    }
}
```

Solution de l’exercice

```
#include <stdio.h>
int resoudre1(int a, int b, float *x);
int main()
{
    int a,b;
    float x;
    int res;
    printf(“Donner a et b de ax+b \n”);
    scanf(“%d %d”,&a,&b);
    res=resoudre1(a,b,&x);
    if(res==0) printf(“Pas de resultat de %d X+ %d “,a,b);
```

```
    else if(res==-1) printf(“les solutions de %d X+ %d : X appartient a R”,a,b);
    else (“Unique resultat de %d X+ %d est %.3f”,a,b,x);
    return 0;
}
int resoudre1(int a, int b, float *x)
{
    if ((a==b)&&(b!=0)) return 0;
    if((a==0)&&(b==0)) return -1;
    *x=-(float)b/a;
    return 1;
}
```