TP2 : Tableaux avec le langage C

septembre

1 Manipulations élémentaires de tableaux unidimensionnels en C

▷ **Question 1.** On considère un tableau de taille n dont chaque case d'indice i correspond au solde bancaire d'une personne numérotée i. Ecrire une fonction de signature :
`void evol_soldes(int tab[],int taille, int icase, int evol )` qui prend en entrée un tel tableau et sa taille, un indice et un entier positif correspondant à un crédit ou négatif s'il s'agit d'un débit. La fonction devra modifier le tableau en tenant compte du débit ou du crédit passé en entrée. Utiliser cette fonction afin d'écrire un programme qui lira au clavier un nombre de personnes n puis un nombre d'opérations m et enfin m couples d'entiers correspondant au numéro de l'individu dont le compte va subir une opération ainsi que de la valeur (relative) de la transaction. Le programme devra afficher les soldes des comptes à l'issue des m opérations pour des personnes disposant chacune initialement de 100 euros. ◁

▷ **Question 2.** On demande à n personnes de choisir un entier entre 1 et 100 et on met les résultats obtenus dans un tableau de taille n . Ecrire une fonction de signature :
`void choix (int tab_pers[] , int tab_choix[] , int nb_pers)` qui prend en entrée un tableau de taille `nb_pers` dont la case i contient la valeur choisie par l'individu i. On veut mettre à jour le tableau `tab_choix` de telle sorte que dans la case k de ce tableau se trouve le nombre de personnes ayant choisies la valeur k. ◁

▷ **Question 3.** Ecrire une fonction qui prend en entrée un tableau d'entiers et renvoie l'élément de valeur minimale dans ce tableau.
La signature de cette fonction sera donc : `int minimum (int tab[], int taille)` ◁

▷ **Question 4.** Ecrire une fonction qui prend en entrée un tableau d'entiers et renvoie la somme de ses éléments.
La signature de cette fonction sera donc : `int somme (int tab[], int taille)` ◁

▷ **Question 5.** Ecrire une fonction de signature `int trié (int tab[],int taille)` qui prend en entrée un tableau et renvoie 1 si celui-ci est trié en ordre croissant et 0 sinon. ◁

▷ **Question 6.** Ecrire une fonction de signature `int distincts (int tab[],int taille)` qui prend en entrée un tableau et renvoie 1 si tous les éléments qu'il contient sont distincts et 0 sinon. ◁

▷ **Question 7.** Créer une fonction `copier` qui prend en paramètre deux tableaux d'entiers. Le contenu du premier tableau devra être copié dans le second tableau.
La signature de cette fonction sera donc :
`void copier(int tableauOriginal[], int tableauCopie[], int tailleTableau)` ◁

▷ **Question 8.** Ecrire une fonction qui prend en entrée trois tableaux et concatène les deux premiers tableaux passés en argument dans le troisième.
La signature de cette fonction sera donc :
`void concatene(int tableau1[], int tableau2[], int tableaures[], int taille1, int taille2)` ◁

2 TABLEAUX À DEUX DIMENSIONS :

Pour déclarer un tableau à deux dimensions on peut par exemple utiliser la notation suivante :

```
int tab[2][3]={{1,2,3},{4,5,6}};
```

▷ **Question 9.** Ecrire une fonction qui prend en entrée un tableau à deux dimensions et qui affiche ses éléments ligne par ligne.

La signature de cette fonction sera donc :

```
void affiche_2D(int tableau[][], int taille_lignes, int taille_col)
```

Normalement vous devriez rencontrer un problème de compilation avec cette fonction. Pour le résoudre, il faut définir une variable globale `nb_colonnes` et travailler avec.

Afin d'éviter de devoir déclarer au préalable le nombre de colonnes de notre tableau, nous allons utiliser des représentations des tableaux 2D légèrement différentes en utilisant des pointeurs. Nous pourrons donc considérer les fonctions suivantes :

```
void affichep_2D(int* tab[],int taille_lignes,int taille_col){
    for (int i = 0;i<taille_lignes;i++){
         for (int j=0;j<taille_col;j++){
              printf("%d",tab[i][j]);
         }
         printf("\n");
    }
}


void affichepp_2D(int** tab,int taille_lignes,int taille_col){
    for (int i = 0;i<taille_lignes;i++){
         for (int j=0;j<taille_col;j++){
              printf("%d",tab[i][j]);
         }
         printf("\n");
    }
}
```

Initialiser un tableau de type `int* tabp[]` et un tableau `int** tabpp` contenant les mêmes éléments que tab. ◁

▷ **Question 10.** Ecrire une fonction qui prend en entrée deux tableaux à deux dimensions et qui transforme le second tableau en la transposée du premier.

La signature de cette fonction sera :

```
void transposep(int** tableau1, int** tableau2, int taille_lignes, int taille_col)
```
◁

3 TABLEAUX ET POINTEURS-REPRÉSENTATION MÉMOIRE DES TABLEAUX

▷ **Question 11.** Ecrire une fonction `int** creer_tableau(int n,int m` qui renvoie un tableau de taille $n * m$ tel que la case (i, j) contient la valeur $i * j$. ◁

▷ **Question 12.** Ecrire une procédure qui crée un tableau triangulaire à deux dimensions dont l'élément d'indice i, j vaut le coefficient binomial $\binom{i}{j}$ avec $0 \leq i \leq n, 0 \leq j \leq i$ où n est saisi au clavier par l'utilisateur. ◁

▷ **Question 13.** Créer un tableau de taille n où n est entré au clavier. Initialiser ce tableau avec des valeurs aléatoires (en prenant le reste du résultat de la fonction `rand`). Créer un tableau de taille $n+1$ correspondant au tableau précédent auquel on a ajouté un élément saisi au clavier. Remplacer enfin le premier tableau par le nouveau de taille $n+1$. ◁

4 Quelques manipulations de tris

▷ **Question 14.** Ecrire une fonction `place` qui prend en entrée un tableau, un numéro de case i et la valeur de l'élément dans la i-eme case et renvoie la place où insérer cet élément dans le tableau qui est supposé trié jusqu'à la case $i-1$ afin qu'il le soit jusqu' à la case i.

La signature de cette fonction sera : `int place (int tab[], int taille, int i, int elt)`

En utilisant judicieusement la fonction `place`, écrire une procédure qui trie un tableau (ce tri est appelé tri par insertion).

On écrira ensuite la procédure similaire suivante : on veut trier un tableau (avec l'algo de tri par insertion), tout en gardant une trace des positions initiales des éléments.

Ainsi, avec le tableau `{88,62,47,16,87,27,61,36,2,40}` la procédure affichera :

```
La 1e valeur vaut 2 et correspond à l'elt d'indice 8
La 2e valeur vaut 16 et correspond à l'elt d'indice 3
La 3e valeur vaut 27 et correspond à l'elt d'indice 5
La 4e valeur vaut 36 et correspond à l'elt d'indice 7
La 5e valeur vaut 40 et correspond à l'elt d'indice 9
La 6e valeur vaut 47 et correspond à l'elt d'indice 2
La 7e valeur vaut 61 et correspond à l'elt d'indice 6
La 8e valeur vaut 62 et correspond à l'elt d'indice 1
La 9e valeur vaut 87 et correspond à l'elt d'indice 4
La 10e valeur vaut 88 et correspond à l'elt d'indice 0
```

◁

▷ **Question 15.**

Ecrire une fonction qui prend en entrée un tableau d'entiers et l'organise de la manière suivante : tous les éléments de valeurs paires sont au début du tableau et ceux de valeurs impaires à la fin.

Par exemple `{12,5,3,6,9,11,4}` sera transformé en `{12,4,6,3,9,11,5}`

La signature de cette fonction sera : `void partage(int tab[], int taille)` ◁

▷ **Question 16.** Ecrire une fonction qui prend en entrée un tableau contenant uniquement les valeurs $1, 2$ et 3 et qui le réorganise pour que le tableau ne contienne que des 1 puis que des 2 puis que des 3. ◁