

LANGAGE C – TP 6 et 7

Pointeurs

Les codes proposés devront impérativement avoir la structure définie dans la fiche de TP1 et des commentaires pertinents seront ajoutés. Une grande importance doit être apportée à la lisibilité des codes.

Exercice 1

1) On s'intéresse au programme suivant :

1	/* TP6 - Pointeurs TD
2	SP - 08/21 - Exemple à analyser (question 1) */
3	/*-----*/
4	/* IMPORTATION BIBLIOTHEQUES */
5	/*-----*/
6	# include <stdio.h>
7	/*-----*/
8	/* PROGRAMME PRINCIPAL */
9	/*-----*/
10	void main()
11	{
12	int x, y, *p1 = NULL *p2 = NULL;
13	x = 3;
14	p1 = &x;
15	y = *p1;
16	y = 5;
17	p2 = &y;
18	(*p1)++;
19	y = p1;
20	p2 = 12;
21	p1 = y;
22	}

Lors de la compilation, les messages d'erreurs suivants sont affichés :

Liste d'erreurs		
<div> 6 erreurs 0 avertissements 0 messages </div>		
	Description	Fichier
1	error C2440: '=' : impossible de convertir de 'int *' en 'int'	extd_analyse.cpp
2	error C2440: '=' : impossible de convertir de 'int' en 'int *'	extd_analyse.cpp
3	error C2440: '=' : impossible de convertir de 'int' en 'int *'	extd_analyse.cpp
4	IntelliSense : impossible d'assigner une valeur de type "int *" à une entité de type "int"	ExTD_analyse.cpp
5	IntelliSense : impossible d'assigner une valeur de type "int" à une entité de type "int *"	ExTD_analyse.cpp
6	IntelliSense : impossible d'assigner une valeur de type "int" à une entité de type "int *"	ExTD_analyse.cpp

- Expliquer ces messages d'erreurs.
- Proposer une correction pour les lignes incriminées.

2) On s'intéresse au programme suivant :

1	/* TP6 - Pointeurs TD
2	SP - 08/21 - Exemple à analyser (question 2) */
3	
4	/*-----*/
5	/* IMPORTATION BIBLIOTHEQUES */
6	/*-----*/
7	
8	# include <stdio.h>
9	
10	/*-----*/
11	/* PROGRAMME PRINCIPAL */
12	/*-----*/
13	
14	void main()
15	{
16	int x, y, *p1 = NULL, *p2 = NULL;
17	x = 3;
18	p1 = &x;
19	y = *p1;
20	y = 5;
21	p2 = &y;
22	(*p1)++;
23	y = *p1;
24	*p2 = 12;
25	*p1 = y;
26	printf("\n\n");
27	}

Indiquer la valeur des variables x, y, p1, p2, *p1 et *p2 lors de l'exécution de chaque ligne du processus (à partir de la ligne 16). On donne : &x : 11532164 &y : 11532152

3) On s'intéresse au code suivant :

1	/* TP6 - Pointeurs TD
2	SP - 08/21 - Exemple à analyser (question 3) */
3	/*-----*/
4	/* IMPORTATION BIBLIOTHEQUES */
5	/*-----*/
6	# include <stdlib.h>
7	# include <stdio.h>
8	/*-----*/
9	/* PROTOTYPES */
10	/*-----*/
11	
12	void fonction1(int *ptr, int taille, int k);
13	void fonction2(int *ptr, int taille);
14	

15	/*-----*/
16	/* FONCTIONS */
17	/*-----*/
18	
19	void fonction1(int *ptr, int taille, int k)
20	{
21	for (int i = 0; i < taille; i++)
22	{
23	// A COMPLETER
24	// A COMPLETER
25	}
26	}
27	
28	void fonction2(int *ptr, int taille)
29	{
30	for (int i = 0; i < taille; i++)
31	{
32	// A COMPLETER
33	// A COMPLETER
34	}
35	}
36	
37	
38	/*-----*/
39	/* PROGRAMME PRINCIPAL */
40	/*-----*/
41	
42	void main()
43	{
44	//Déclaration des variables
45	int tableau[3] = { 2, 4, 6 };
46	int x=2, n=3;
47	int* p = NULL;
48	p = &tableau[0];
49	
50	// Tableau initial
51	printf(" Tableau initial : [");
52	for (int i = 0; i < 3; i++)
53	{
54	printf(" %d ", tableau[i]);
55	}
56	printf("]\n");
57	
58	// Traitement du tableau
59	Fonction1(p, n, x);
60	printf(" Apres fonction1 : [");
61	for (int i = 0; i < 3; i++)
62	{
63	// A COMPLETER
64	}
65	printf("]\n");
66	Fonction2(p, n);
67	printf(" Apres fonction2 : [");

68	<code>for (int i = 0; i < 3; i++)</code>
69	<code>{</code>
70	<code>// A COMPLETER</code>
71	<code>}</code>
72	<code>printf("]\n");</code>
73	<code>printf("\n\n");</code>
74	<code>}</code>

Compléter les fonctions fonction1 et fonction2, ainsi que les lignes 63 et 70, qui devront utiliser la variable p, pour obtenir la sortie suivante :

```
Tableau initial : [ 2  4  6  ]
Après fonction1 : [ 4  8 12  ]
Après fonction2 : [16 64 144 ]
```

Exercice 2

Ecrire un programme demandant à l'utilisateur un nombre de secondes sous forme d'un entier est qui convertit ce nombre en nombre de jours, d'heures, de minutes et de secondes. La conversion devra être faite par une fonction appelée depuis le programme principal.

Deux solutions seront programmées : la première utilisera des pointeurs sur les différentes variables issues de la conversion, l'autre stockera ces variables dans un tableau.

La sortie écran devra suivre le modèle suivant :

```
Nombre de secondes a convertir : 75287
PREMIERE SOLUTION : 4 POINTEURS
-----
nbre de jours : 0
nbre d heures : 20
nbre de minutes : 54
nbre de secondes : 47

SECONDE SOLUTION : TABLEAU
-----
nbre de jours : 0
nbre d heures : 20
nbre de minutes : 54
nbre de secondes : 47
```

Exercice 3

On s'intéresse au code suivant :

```
/* TP6 - POINTEURS
SP - 08/21 - Exercice 3 */

/*-----*/
/* IMPORTATION BIBLIOTHEQUES */
/*-----*/

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

/*-----*/
/* PROGRAMME PRINCIPAL */
```

```
/*-----*/  
  
void main()  
{  
    int** matrice[4];  
    int* ligne1[6];  
    int* ligne2[6];  
    int* ligne3[6];  
    int* ligne4[6];  
    for (int i = 0; i < 6; i++)  
    {  
        *(ligne1 + i) = i;  
    }  
    for (int i = 0; i < 6; i++)  
    {  
        *(ligne2 + i) = i*2;  
    }  
    for (int i = 0; i < 6; i++)  
    {  
        *(ligne3 + i) = i*3;  
    }  
    for (int i = 0; i < 6; i++)  
    {  
        *(ligne4 + i) = i*4;  
    }  
    matrice[0] = ligne1;  
    matrice[1] = ligne2;  
    matrice[2] = ligne3;  
    matrice[3] = ligne4;  
    for (int i = 0; i < 6; i++)  
    {  
        printf ("%d\t", *(ligne1 + i));  
    }  
    printf("\n");  
    for (int i = 0; i < 6; i++)  
    {  
        printf ("%d\t", *(ligne2 + i));  
    }  
    printf("\n");  
    for (int i = 0; i < 6; i++)  
    {  
        printf ("%d\t", *(ligne3 + i));  
    }  
    printf("\n");  
    for (int i = 0; i < 6; i++)  
    {  
        printf ("%d\t", *(ligne4 + i));  
    }  
    printf("\n\n");  
}
```

Indiquer quelle sera la sortie écran.

Exercice 4

On souhaite écrire un code qui bâtit une matrice 4x4 à l'aide d'entiers aléatoires pris entre 0 et 100, affiche cette matrice, la transpose puis affiche la transposée de cette matrice.

Une matrice peut être modélisée comme un tableau à deux dimensions.

$$\text{Ainsi, la matrice } M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

est stockée sous la forme : $M = [[1,2,3,4] , [5,6,7,8] , [9,10,11,12] , [13,14,15,16]]$.

La transposée de la matrice M , notée M^t est définie par : $M^t[i][j] = M[j][i]$, soit, pour la

$$\text{matrice } M \text{ définie ci-dessus : } M^t = \begin{bmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{bmatrix}.$$

Dans cet exercice, on souhaite utiliser les pointeurs pour accéder aux éléments de la matrice. Ainsi, la syntaxe $M[i][j]$ ne sera pas utilisée.

- 1) Importer les bibliothèques `stdlib` et `stdio`. Définir deux constantes globales `mini` et `maxi` valant respectivement 0 et 100. Ces deux constantes seront de type `int`.
- 2) Définir dans le `main()` une variable nommée `matrice` qui sera un tableau d'`int` de 4x4 éléments. Définir un pointeur nommé `pointeur` qui pointe sur le premier élément de `matrice`.
- 3) Ecrire une fonction : `void remplitMatrice (int* ptr)` qui remplit le tableau `matrice` via le pointeur passé en paramètre avec des entiers aléatoires. Pour cela, utiliser : `(rand() % (maxi - mini + 1) + mini)`.
- 4) Ecrire une fonction : `void afficheMatrice (int* ptr)` qui affiche la matrice pointée par le pointeur passé en paramètre. Cet affichage devra correspondre à celui d'une matrice 4x4 (voir sortie écran attendue).
- 5) Ecrire une fonction : `void transposeMatrice (int* ptr)` qui modifie la matrice pointée par le pointeur passé en paramètre pour la transposer.
- 6) Compléter le programme principal pour générer l'affichage suivant :

```
Matrice initiale <pointeur>
41      85      72      38
80      69      65      68
96      22      49      67
51      61      63      87

Matrice transposee <pointeur>
41      80      96      51
85      69      22      61
72      65      49      63
38      68      67      87
```

Exercice 5

On souhaite écrire un programme permettant de déterminer le barycentre d'un nuage de points. Le nombre de points sera acquis par l'utilisateur et se verront attribuer des valeurs aléatoires comprises entre 0 et 100. Un type personnalisé, associé à une structure, sera utilisé.

La sortie écran attendue est la suivante :

```
Nombre de points souhaites : 10
Points crees
-----
point 0 : 41.00 5.00
point 1 : 10.00 31.00
point 2 : 44.00 16.00
point 3 : 3.00 33.00
point 4 : 34.00 35.00
point 5 : 44.00 44.00
point 6 : 25.00 48.00
point 7 : 16.00 32.00
point 8 : 37.00 8.00
point 9 : 33.00 30.00

Barycentre des points
-----
Coordonnees : 28.70      28.20
```

- 1) Définir un type nommé point, associé à une structure nommée pt comprenant deux champs : un float x et un float y, correspondant aux coordonnées du point.
- 2) Importer les bibliothèques stdio, stdlib et assert. Définir deux constantes globales mini et maxi, de valeurs respectives 0 et 100, de type int.
- 3) Ecrire, dans le programme principal, les instructions permettant d'acquérir auprès de l'utilisateur le nombre de points composant le nuage.
- 4) Ecrire dans le programme principal les instructions permettant l'allocation mémoire pour stocker le nuage de points.
- 5) Ecrire une fonction de signature `void rempliPt(point* p, int n);` qui affecte les valeurs aléatoires aux champs des n points constituant le nuage. Ces valeurs seront comprises entre mini et maxi. On rappelle que l'instruction `rand()` renvoie un int. L'accès à un champs de structure lorsque la structure est accessible via son adresse (donc un pointeur) se fait avec la syntaxe : `(pointeur)-> champs`.
- 6) Ecrire, dans le programme principal, les instructions permettant d'afficher les coordonnées des points ainsi créés. L'affichage des float sera limité à 2 chiffres après la virgule (`%.2f`).
- 7) Ecrire une fonction de signature `point rechercheBarycentre(point* p, int n);` permettant la création du barycentre du nuage de n points.
- 8) Ecrire, dans le programme principal les instructions affichant les coordonnées du barycentre du nuage. L'affichage des float sera limité à 2 chiffres après la virgule (`%.2f`).