Langage C TP 5

I) Les variables dévoilent pour nous leurs adresses – L'opérateur &

1. Qu'observe-t-on lors de l'exécution de ce programme ?

L'adresse de chaque case est égale à celle de la précédente + 4

2. Illustrez sous forme d'un schéma (plan mémoire) similaire à celui de la Figure 1 la représentation en mémoire du tableau « t ».

Adresse	Mémoire
X	t[0]=10
X+4	t[0]=20
X+8	t[0]=30
X+12	t[0]=40
X+16	t[0]=50

3. Compilez et exécutez plusieurs fois ce programme en remplaçant dans la déclaration du tableau « t » le type « int » par « short int », « float », « double ». Que peut-on en déduire sur le format de stockage des types de données mentionnés ?

Un int est stocké sur 4 octets,un short int sur 2 octets, un float sur 4 octets et un double sur 8 octets.

4. Écrire le programme suivant et expliquer à l'aide d'un schéma son fonctionnement et sa représentation mémoire.

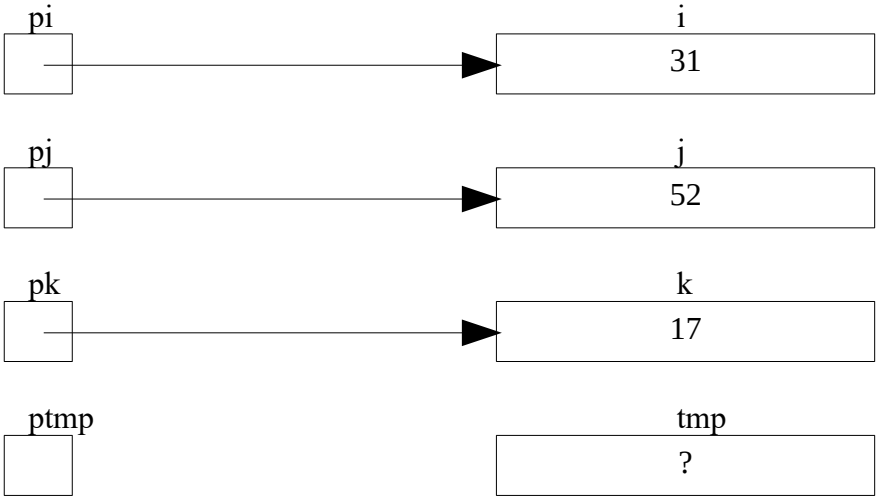
```
Adresse de px : FFFFCC28
Adresse pointee par le pointeur px = 28460
Valeur memorisee a l'adresse pointee par le pointeur px = 1
```

Adresse	Mémoire
FFFFCC28	28460
28460	1

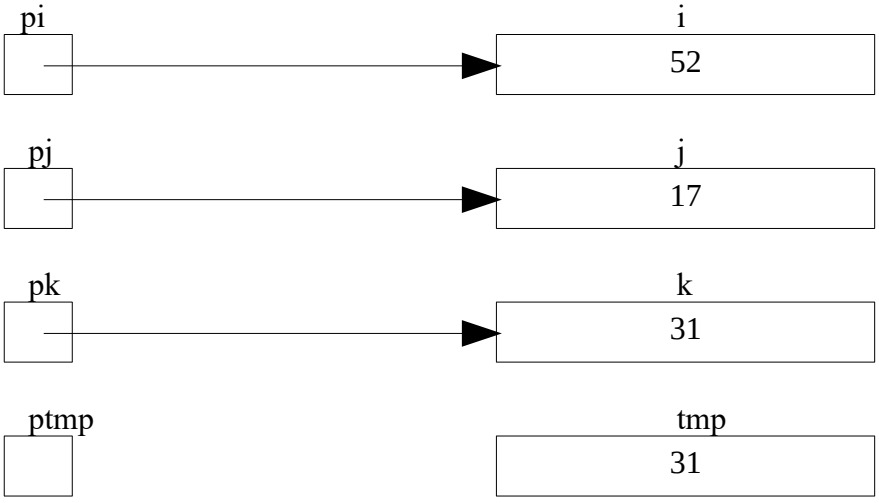
II) Accéder au contenu des variables de manière indirecte: L'opérateur *

Représentez schématiquement les étapes successives de l'exécution du programme suivant.

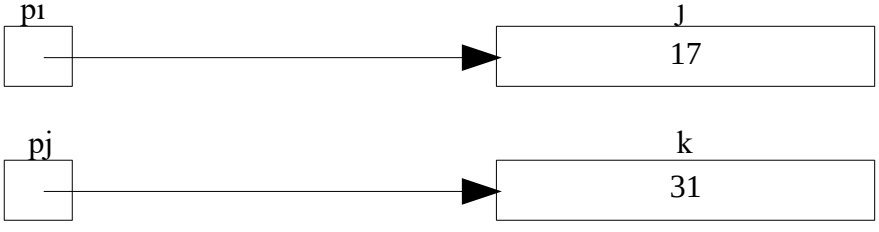
1) Création des variables entières i,j,k et tmp et des pointeurs pi, pj, pk et ptmp. Affectation des valeurs 31, 52 et 17 aux variables i, j et k. L'adresse des variables i, j et k sont affectées aux pointeurs pi, pj et pk.

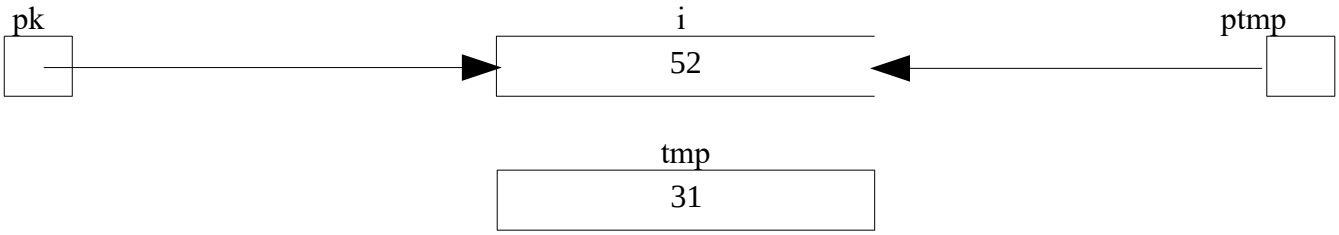


2)Affectation de la valeur à l'adresse pointée par pi (i) dans tmp, de la valeur à l'adresse pointée par pj (j) à l'adresse pointée par pi (i), de la valeur à l'adresse pointée par pk (k) à l'adresse pointée par pj (j) puis de la valeur de tmp à l'adresse pointée par pk (k).



3)Affectation de la valeur de pi à ptmp, de la valeur de pj à pi, de la valeur de pk à pj puis de la valeur de ptmp à pk.

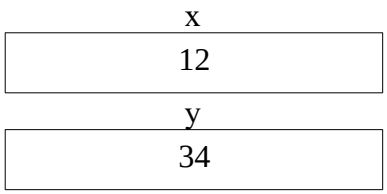




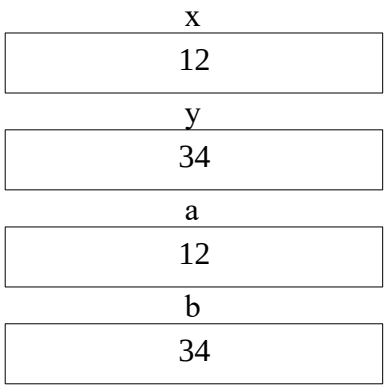
III) Écrire une procédure qui modifie la valeur de ses arguments

1. Expliquer, à l'aide de schémas et en détaillant les étapes de l'exécution, pourquoi le programme ci-dessus ne fonctionne pas.

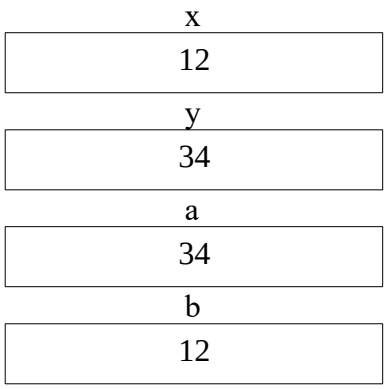
1)Création des variables x et y, et affectations des valeurs 12 et 34.



2) Appel de la fonction echanger() : les variables x et y sont passées par valeur, et deux nouvelles variables a et b sont créées.

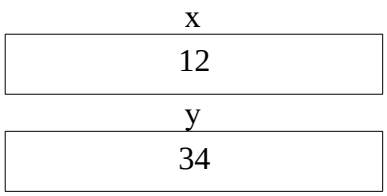


3)Les valeurs de a et b sont échangées, et non celles de x et y

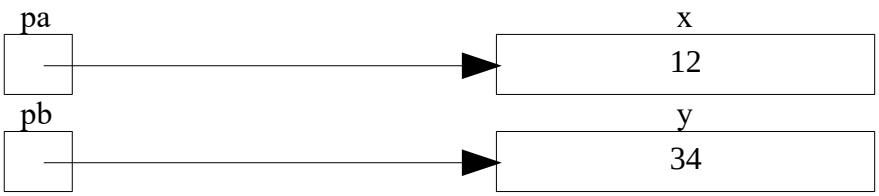


3. Expliquer, à l'aide de schémas et en détaillant les étapes de l'exécution, pourquoi le programme corrigé fonctionne.

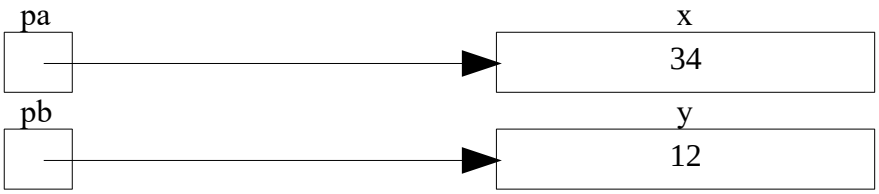
1)Création des variables x et y, et affectations des valeurs 12 et 34.



2) Appel de la fonction echanger() : Les pointeurs pa et pb sont créés, ils pointent sur les variables x et y



3)Les valeurs aux adresses pointées par pa et pb sont échangées

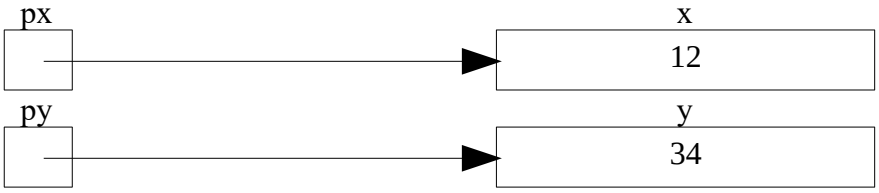


IV) Introduire des doubles pointeurs

Expliquer le fonctionnement de la procédure suivante. Écrire une fonction « main » similaire

à celle proposée dans III), et qui illustre l'utilisation de cette procédure.

1)Création des variables x et y, affectations des valeurs 12 et 34, et création des pointeurs px et py pointant sur leurs adresses.



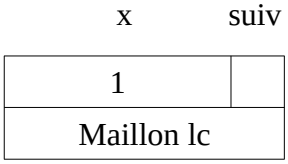
2) Appel de la fonction echangerPointeurs() : Les doubles pointeurs ppa et ppb sont créés, ils pointent sur les pointeurs px et py

3)Les valeurs aux adresses pointées par ppa et ppb sont échangées. On échange donc les adresses pointées par px et py

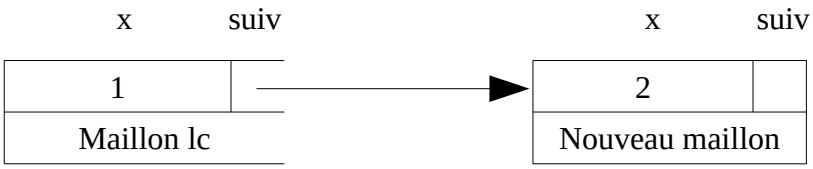
V) met en application la notion de pointeur – Listes chaînées

1. Écrire le programme suivant et expliquer à l'aide d'un schéma son fonctionnement et sa représentation mémoire.

1) Création du maillon lc, et affectation de la valeur 1 au maillon (attribut x)

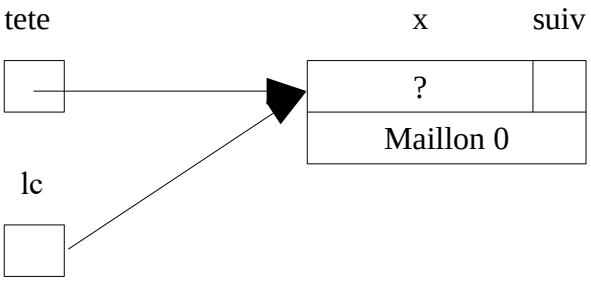


2) Affectation de l'adresse d'un nouveau maillon à l'attribut suiv de lc. On affecte la valeur 2 à l'attribut x de ce nouveau maillon.

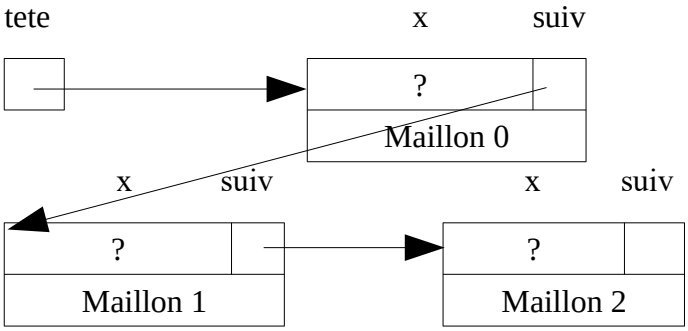


2. Ecrire le programme suivant et expliquer à l'aide d'un schéma son fonctionnement et sa représentation mémoire.

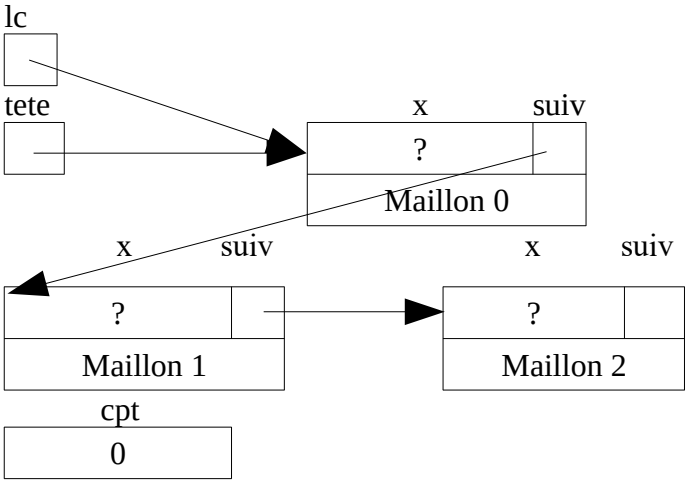
1) Création des pointeurs tete et lc vers des maillons. On fait pointer lc et tete sur un nouveau maillon



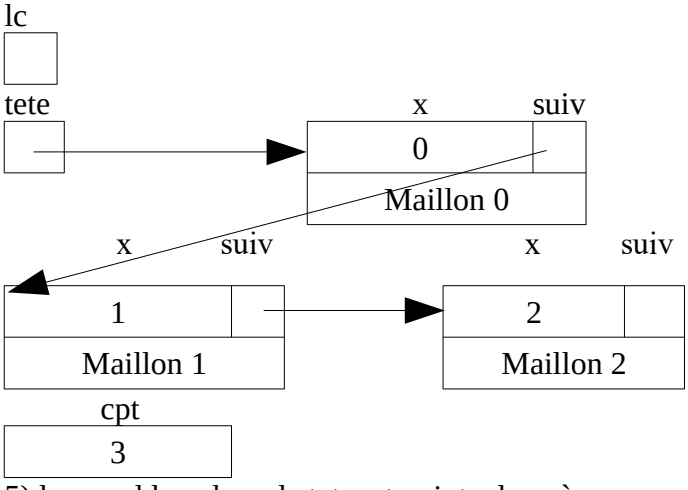
2) Création de 2 autres maillons. Pour ce faire, on crée un maillon qui sera pointé par l'attribut suiv du maillon pointé par lc, puis on fait pointer lc sur ce nouveau maillon. A la fin lc a la valeur de l'attribut suiv du dernier maillon, c'est à dire NULL.



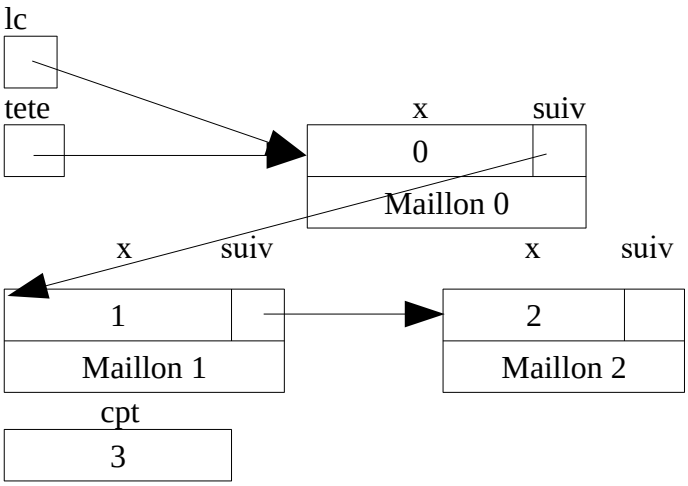
3) lc prend la valeur de tete, et pointe donc à nouveau sur le maillon 0. On crée un compteur cpt initialisé à 0



4) On assigne la valeur de cpt à l'attribut x du maillon pointé par lc, puis on incrémente cpt et lc prend la valeur de l'attribut suiv du maillon qu'il pointe. lc pointe alors vers le maillon suivant. On recommence l'opération jusqu'à ce que lc soit égal à NULL, c'est à dire qu'il ait atteint le dernier maillon



5) lc prend la valeur de tete, et pointe donc à nouveau sur le maillon 0.



6) On affiche la valeur de l'attribut x du maillon pointé par lc, l'adresse du maillon courant c'est à dire la valeur de lc, et l'adresse du maillon suivant, c'est à dire la valeur de l'attribut suiv du maillon pointé par lc. On affecte ensuite à lc cette valeur, et on recommence tant qu'il n'est pas égal à NULL, c'est à dire qu'on ait parcouru tous les maillons