Travaux Pratiques N° 2

Les Pointeurs, Allocation Dynamique de Mémoire

EXERCICE 1 : Adresses & Tableaux

Un programme contient la déclaration suivante:

int tab[10] = {4,12,53,19,11,60,24,12,89,19};

Compléter ce programme de sorte d'afficher les adresses des éléments du tableau.

EXERCICE 2: Pointeurs, Arithmétique & Tableau

Soit iptr un pointeur qui 'pointe' sur un tableau tabA.

Quelles valeurs ou adresses fournissent ces expressions :

1. *iptr+2,
2. *(iptr+2),
3. &iptr+1 ,
4. &tabA[4]-3 ,
5. tabA+3 ,
6. &tabA[7]-iptr
7. iptr+(*iptr-10) ,
8. *(iptr+*(iptr+8)-tabA[7]),

EXERCICE 3 : Pointeurs, Longueur d'une chaine de caractères

Écrire un programme qui lit une chaîne de caractères CH de taille maximum 100 et détermine la longueur de la chaîne à l'aide d'un pointeur P. Le programme n'utilisera pas de variables numériques.

EXERCICE 4 : Pointeurs, Palindrome

Un palindrome est un mot qui reste le même qu'on le lise de gauche à droite ou de droite à gauche (par exemple, PIERRE n'est pas un palindrome, alors que OTTO est un palindrome). Ecrire de deux façons différentes, un programme qui vérifie si une chaîne CH introduite au clavier est un palindrome.

i. En utilisant uniquement le formalisme tableau.
ii. En utilisant des pointeurs au lieu des indices numériques.

EXERCICE 5 : Arithmétique des pointeurs

Etant donné le programme ci-dessus :

```
int  main()
{
  // init
  int a = 1;
  int b = 2;
  int c = 3;
  int *p1, *p2;

  p1=&a;
  p2=&c;
  *p1 = *p2;
  (*p2)++;
  p1=p2;
  p2=&b;
  *p1-=*p2;
  ++(*p2);
  *p1*=*p2;
  ++(*p2);
  a=*p2**p1;
  p1=&a;
  *p2=*p1/=*p2;
  return 0;
}
```

Compléter le tableau suivant en se basant sur les instructions du programme ci-dessous :

	a	b	c	p1	p2
init	1	2	3	?	?
p1=&a	1	2	3	&a	?
p2=&c;					
*p1=*p2;					
(*p2)++;					
p1=p2;					
p2=&b;					
*p1-=*p2;					
++(*p2);					
p1=*p2;					
p1=&a;					
*p2=*p1/=*p2;					

EXERCICE 6 : Pointeurs, chaine de caractères

Ecrire un programme C qui lit une chaine de caractères et affiche cette chaine à partir de la première occurrence d'un caractère entré par l'utilisateur en utilisant la fonction strchr et un pointeur pour le parcours de la chaine.

NB : `char * strchr( const char * string, int searchedChar );`

Paramètres:

- `string` : la chaîne de caractères dans laquelle effectuer la recherche.
- `searchedChar` : permet de définir le caractère recherché.

Valeur de retour:

Soit le caractère recherché est trouvé et dans ce cas un pointeur sur la position du caractère recherché vous sera retourné. Soit le caractère n'est pas présent dans la chaîne et dans ce cas, un pointeur nul vous sera renvoyé.

EXERCICE 7 : Pointeurs, Recherche & Compression d'un tableau

Écrire un programme qui lit un entier x et un tableau tabA du type int au clavier et élimine toutes les occurrences de x dans tabA en tassant les éléments restants. Le programme utilisera les pointeurs p1 et p2 pour parcourir le tableau.

EXERCICE 8 : Allocation dynamique de mémoire

Ecrire la fonction qui alloue la mémoire d'un vecteur de taille dimension, puis qui l'initialise à la valeur val, « *int * alloue_vecteur(int dimension, int val)* ». Ecrire la fonction « *void libere_vecteur(int * vecteur)* », qui libère le vecteur vecteur. Afficher ce vecteur pour tester vos fonctions.

EXERCICE 9 : Pointeurs, Tableaux à deux dimensions

Écrire un programme qui lit 5 mots d'une longueur maximale de 50 caractères et les mémorise dans un tableau de chaînes de caractères TABCH.
Inverser l'ordre des caractères à l'intérieur des 5 mots à l'aide de deux pointeurs P1 et P2.
Afficher ces mots.

EXERCICE 10 : Tableau de pointeurs

Ecrire un programme qui lit le jour, le mois et l'année d'une date au clavier et qui affiche la date en français et en allemand. Utiliser deux tableaux de pointeurs, MFRAN et MDEUT que vous initialisez avec les noms des mois dans les deux langues. La première composante de chaque tableau contiendra un message d'erreur qui sera affiché lors de l'introduction d'une donnée illégale.

Exemples:

```
Introduisez la date: 1 4 1993
Luxembourg, le 1er avril 1993
Luxemburg, den 1. April 1993

Introduisez la date: 2 4 1993
Luxembourg, le 2 avril 1993
Luxemburg, den 2. April 1993
```

EXERCICE 11 : Remplissage d'un tableau avec des valeurs aléatoires

Ecrire une fonction **remplirAlea** permettant de remplir aléatoirement un tableau **tab** de **nb** entiers avec des valeurs tirée au hasard dans l'intervalle [**0**, **max**[.

Pour des raisons d'optimisation, il faudra allouer la mémoire dynamiquement.

EXERCICE 12 : Allocation dynamique de mémoire

Ecrire un programme qui lit 10 phrases d'une longueur maximale de 200 caractères au clavier et qui les mémorise dans un tableau de pointeurs sur char en réservant

dynamiquement l'emplacement en mémoire pour les chaînes. Ensuite, l'ordre des phrases est inversé en modifiant les pointeurs et le tableau résultant est affiché.

EXERCICE 13 : Tri par insertion d'un tableau d'entier

Réalisez un programme qui effectue le tri "par insertion" d'un tableau de 10 entiers (que vous aurez initialisé à la déclaration).

La méthode du tri par insertion est celle qu'utilise un joueur de carte lorsqu'il trie les cartes qu'il a reçu. Il prend la première carte et la compare à toutes les suivantes. Si il rencontre une carte plus petite, il la place avant la carte de référence (toutes les cartes entre la première et celle plus petite sont alors décalées). Il répète cette opération jusqu'à la dernière carteLa fonction qui réalise le tri doit traiter 2 paramètres: le nombre d'éléments et le nom du tableau.

EXERCICE 14 : Suppression selon un ordre lexicographique

Ecrire un programme qui lit 10 mots au clavier (longueur maximale: 50 caractères) et attribue leurs adresses à un tableau de pointeurs MOT. Effacer les 10 mots un à un, en suivant l'ordre lexicographique et en libérant leur espace en mémoire. Afficher à chaque fois les mots restants en attendant la confirmation de l'utilisateur (par 'Enter').

EXERCICE 15 : Fusion de deux tableaux

Écrire une fonction int ***fusion(int n1, int *tab1, int n2, int *tab2)*** qui alloue la mémoire nécessaire à un tableau de taille n1 + n2, y copie dans l'ordre croissant les éléments de tab1 et tab2 (on suppose que tab1 et tab2 sont déjà triés) et renvoie l'adresse de ce nouveau tableau.

EXERCICE 16 : Le triangle de Pascal

En mathématiques, le triangle de Pascal est un arrangement géométrique qui stocke les coefficients du developpement de $(x+y)^i$ qui sont les coefficients binomiaux $\binom{i}{j}$. À la lignes i et colonnes j $(0 \leqslant j \leqslant i)$ est placé le coefficient binomial $\binom{i}{j}$, dont voici la représentation : (vous trouverez plus de detail sur la page `http://fr.wikipedia.org/wiki/Triangle_de_Pascal`)

1									
1	1								
1	2	1							
1	3	3	1						
1	4	6	4	1					
1	5	10	10	5	1				
1	6	15	20	15	6	1			
1	7	21	35	35	21	7	1		
1	8	28	56	70	56	28	8	1	
1	9	36	84	126	126	84	36	9	1
1	10	45	120	210	252	210	120	45	10	1

1) Ecrire la fonction qui alloue la mémoire d'une matrice triangulaire inférieure carrée « `int ** alloue_matrice_pascal(int dimension)` ».
2) Ecrire une foction « `int ** remplit_matrice_pascal(int dimension)` » qui stocke les coefficients binomiaux d'un polynôme de taille n (matrice de Pascal de taille n).
3) Ecrire une fonction « `void affiche_matrice_pascal(int dimension)` » qui affiche une matrice de pascal de taille n.