TD 2 : allocation dynamique

François Delbot
Miage L2 - Université de Nanterre

14 octobre 2011

Chaque fois que vous implémentez une fonction, pensez à bien indenter votre code, à le commenter (sans pour autant paraphraser) et à le standardiser ...

Exercice 1. Pour se familiariser avec les pointeurs. Ecrire un programme qui :

1. Initialise des variables a,b,c de type entier, d de type caractère, e de type flottant (double ou float).
2. Déclare des pointeurs (du bon type) p sur b, q sur d et r sur e.
3. Imprime à l'écran, pour chaque pointeur, la valeur du pointeur (l'adresse de l'objet pointé) et la valeur de l'objet pointé.

Faire maintenant pointer p à c et imprimer son adresse et la valeur de l'objet pointé. Modifier la valeur pointée par p pour qu'elle soit égale à la valeur de a. Imprimer à l'écran l'adresse et la valeur pointée par p, puis les valeurs de a, b et c.

Exercice 2. On s'amuse avec les pointeurs... Compléter le tableau en indiquant les valeurs des différentes variables au terme de chaque instruction du programme suivant (on peut aussi indiquer sur quoi pointent les pointeurs) :

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

Exercice 3.Échange de variables. Écrivez une fonction qui permet d'échanger le contenu de deux variables de type int qui lui sont extérieures. Écrire l'appel de cette fonction dans un main.

Exercice 4. Ecrire une fonction qui alloue dynamiquement un tableau dont la taille est passée en argument et le remplit de 0. Cette fonction doit retourner l'adresse du premier élément du tableau. Ecrire une fonction qui accepte en argument un pointeur vers un tableau ainsi que sa taille et le remplit de valeurs aléatoires. Enfin, écrire une fonction qui affiche les éléments d'un tableau. Utiliser cette dernière pour vérifier que vos deux premières fonctions marchent comme on le souhaite.

Exercice 5. Considérons les déclarations suivantes :
int tab[] = {5, 15, 34, 54, 14, 2, 52, 72} ;
int *p = &tab[1], *q = &tab[5] ;

1. Quelle est la valeur de *(p+3) ?
2. Quelle est la valeur de *(q-3) ?
3. Quelle est la valeur de q-p ?
4. La condition $p < q$ est-elle vraie ou fausse ?
5. La condition $*p < *q$ est-elle vraie ou fausse ?

Exercice 6.Concaténation. Écrivez une fonction qui prend en arguments deux tableaux et leurs tailles respectives, et qui renvoie leur concaténation. Il faudra allouer un nouveau tableau pour contenir cette concaténation.

Exercice 7. Implémenter l'algorithme du tri fusion vu en cours en utilisant l'allocation dynamique.

Exercice 8. Ecrire une fonction qui alloue la mémoire d'une matrice de taille lignes*colonnes, puis qui l'initialise à la valeur val,

```
int ** allouematrice(in t lignes,int colonnes,int val)
```

Ecrire la fonction

```
void liberematrice(int ** matrice, int n)
```

qui libère la matrice. Afficher cette matrice pour tester vos fonctions.

Exercice 9. Le triangle de pascal. Ecrire un programme qui construit le triangle de Pascal de degré n et le mémorise dans une matrice triangulaire Mat allouée dynamiquement.

Exemple : Triangle de Pascal de degré 6 :

n=0	1						
n=1	1	1					
n=2	1	2	1				
n=3	1	3	3	1			
n=4	1	4	6	4	1		
n=5	1	5	10	10	5	1	
n=6	1	6	15	20	15	6	1

Méthode : Calculer et afficher seulement les valeurs jusqu'à la diagonale (incluse). Limiter le degré à entrer par l'utilisateur à 13.

Construire le triangle :
- Initialiser le premier élément et l'élément de la diagonale à 1.
- Calculer les valeurs entre les éléments initialisés de gauche à droite en utilisant la relation :

$$P_{i,j} = P_{i-1,j} + P_{i-1,j-1}$$