Programmation 3

L2 Informatique 2012-2013

Fiche de TD 3

Notions abordées : entrées/sorties ; allocation dynamique ; pointeurs ; coercition.

Exercice 1. (Entrées/sorties)

Qu'affiche le programme suivant?

```
#include <stdio.h>
main() {
        char *chaine = "Wolfgang Amadeus Mozart";
        printf("|%s|\n", chaine);
        printf("|%.16s|\n", chaine);
        printf("|%-23.16s|\n", chaine);
        printf("|%23.16s|\n", chaine);
}
```

Exercice 2. (Pointeurs)

Pour chacune des 3 séries d'instructions suivantes, donner chronologiquement les valeurs des variables présentes sur papier en faisant des schémas.

int a, b, * p;
 	
a = 10;
p = &a;
b = *p + 2;
*p = *p + 4;
*p + = 11;
*p = 0;
	int x, y, * p;
	
	x = 2;
	p = &x;
	++*p;
	(*p)++;
	y = ++*p;
	y = (*p) ++;
	y = *p;
		int x, y;
		int a [10];
		int * pa;
		int * pb;
		
		a[0] = 4;
		a[3] = 2;
		a[7] = 11;
		pa = &a[0];
		x = *pa;
		pb = pa + 3;
		x = *pb;
		y = *(pa + 3);
		x = *(pb + 4);
		x = *pb + 4;

Exercice 3. (Pointeurs)

Dans la série d'instructions suivantes, donner chronologiquement les valeurs des variables présentes sur papier en faisant des schémas.

```
#include <stdio.h>

void main()
{
   char buffer[32] = "Langage C et les pointeurs";
   char *ptr;
   int i = 0;

   buffer[10] = 'a';
   *(buffer + 9) = 'b';
   buffer[i++] = 'c';
   i += 15;
   ptr = buffer + i;
   ptr[0] = 'd';
   *++ptr = 'e';

   printf("buffer = %s \n",buffer);
   printf("ptr = %s \n", ptr);
   return 0;
}
```

Exercice 4. (Pointeurs)

Ecrire une fonction d'initialisation à 0 des valeurs d'un tableau de réels double précision.

Le code de la fonction doit utiliser l'arithmétique des pointeurs.

Ecrire un appel pour un tableau, nommé notes, de 10 double.

Exercice 5. (Pointeurs)

Ecrire une fonction d'échange des valeurs de 2 pointeurs d'entiers ainsi qu'un appel conforme.

Exercice 6. (tableau de pointeurs sur des chaînes de caractères)

Soit un programme contenant les déclarations suivantes :

```
char * tab[] = {
                    "Wolfgang Amadeus Mozart",
                    "Ludwig van Beethoven",
                    "Hector Berlioz",
                    "Nicolo Paganini",
                    NULL
                };

char ** p = tab;
```

tab[0] ou *tab ou p[0] ou *p	pointe sur la chaîne
tab[1] ou *(tab + 1) ou pt[1] ou *(p + 1)	pointe sur la chaîne
tab[2] ou *(tab + 2) ou p[2] ou *(p + 2)	pointe sur la chaîne
tab[3] ou *(tab + 3) ou p[3] ou *(p + 3)	pointe sur la chaîne
tab[0][0] ou **tab ou p[0][0] ou **p	vaut
tab[2][3] ou * ((*(tab+2)) + 3)	vaut
p[2][3] ou * ((*(p+2)) + 3)	vaut

Déterminer la valeur des expressions :

p = tab; (* p++)[1] *p++[1] (*(++p))[4] *++*p	p = tab; * (p++)[1] *p[1]++ (*(++p))[4] *++*p

Exercice 7. (Allocation dynamique)

1) Lire 10 phrases au clavier et mémoriser les phrases en utilisant un tableau de pointeurs sur des chaînes de caractères:

- Besoin déterministe connu dès le départ: un buffer de sasie de taille BUFSIZ
- Besoin déterministe connu dès le départ: le nombre de phrases
- Besoin non déterministe connu pendant l'exécution: la longueur des phrases (allocation dynamique)

2) Libérer la mémoire allouée dynamiquement.

Exercice 8. (Comparaison)

Ecrire une fonction comparer qui permet de comparer deux variables d'un type quelconque dont la valeur est représentée sur n octets. La fonction retourne 1 si et seulement si les deux variables sont égales, 0 sinon

Exercice 9. (Listes chaînées)

On définit les structures de maillon et de liste:

```
struct maillon
{
   int valeur;
   struct maillon * suivant;
};

typedef struct maillon * Liste;
```

Définir les fonctions suivantes:

1) Liste insere_tete(int x, Liste l);
2) Liste supprime_tete(Liste l);
3) void affiche(const Liste l);
4) int longueur(const Liste l);
5) Liste insere_fin(int x, Liste l);
6) int recherche(int x, const Liste l); /* version récursive */