TD01 Algorithmique Avancée
Remise à niveau en C

Remarque importante : La première (et éventuellement la deuxième) séance de TD sera consacrée à une remise à niveau du langage C. L'étudiant est amené à programmer le maximum de solutions de ce TD et de venir la première séance de TP avec le code pour être évalué et noté.

Exercice 01 :

main, compilation, édition de liens, déclarations (constantes, variables), types simples, pré-processeur, code ASCII, commentaires, affectation simple

1. **Ecrire le programme C le plus court possible**
2. **Ajouter au programme précédent la définition de la constante PI en respectant les majuscules *(#define)***
3. **Ajouter la déclaration d'une variable entière dont l'identificateur sera *rayon* (*int*)**
4. **Ajouter les instructions qui permettent d'initialiser la variable rayon avec la valeur 10, puis calculer la surface du cercle en stockant le résultat dans une variable dont l'identificateur sera *resultat*. Cette dernière variable sera de type réel**
5. **Ecrire un programme en C qui saisit deux nombres entiers x et y, les affiche, inverse leur contenu et les affiche de nouveau**
6. **Créer un programme qui déclare un caractère (type char) et l'initialise une première fois avec le caractère '7', puis avec le caractère correspondant au code ASCII 98**

Exercice 2 :

Déclaration de variables structurées de type enregistrement, usage

1. **Détailler avec l'assistant de TD la déclaration d'une structure (enregistrement) en C, ainsi que les deux modes d'accès aux informations d'une structure.**
2. **Ecrire la déclaration d'une structure qu'on appellera Un_Tableau_Entier contenant un tableau de taille NMAX entiers et une variable entière nommée ncase donnant le nombre de cases réellement utilisées dans le tableau ; ncase sera initialisée avec la valeur 0**
3. **Définir le type structuré d'un tonneau, sachant qu'un tonneau étant bombé, il possède un petit diamètre d (les deux bouts), un grand diamètre D (partie ventrue) et une longueur L. Le volume d'un tonneau est donné par la formule suivante :**

$$V = 3.14159\ L \left(\frac{d}{2} + \frac{2}{3}\left(\frac{D}{2} - \frac{d}{2}\right)\right)^2.$$

4. **Enfin un tonneau contient quelque chose (de l'huile d'olive). On utilisera la syntaxe avec typedef.**

Exercice 3 :

Opérateurs "adresse de" et "contenu de"

1. **Ecrire un programme qui déclare deux variables i et j ; la première (i) sera une variable de type entier, la seconde (j) sera une variable de type pointeur sur entier. C'est-à-dire que j contiendra une valeur qui sera l'adresse (emplacement mémoire) d'un entier. Affecter à l'entier i la valeur 5, affecter à l'adresse j l'adresse de l'entier**

i, afficher l'entier i et le contenu de ce qui est pointé par la variable j.

2. **Modifier le programme précédent en ajoutant à la fin les instructions suivantes : incrémenter la donnée pointée par la variable j, afficher i, multiplier i par 5, afficher la donnée pointée par j, incrémenter l'adresse j et afficher de nouveau la donnée pointée par j.**

Exercice 4 : (1 à 5)

Déclaration de variables structurées de type tableau, initialisation de tableau à la déclaration, liens entre tableau et adresse

1. **Ecrire un programme où on déclarera un tableau de 10 entiers, une variable de contrôle de boucle de type entier ; puis on réalisera l'initialisation du tableau avec la valeur 7 à l'aide d'une boucle**
2. **Déclarer un tableau de 5 entiers en l'initialisant à la déclaration avec les valeurs décroissantes allant de 4 à 0**
3. **Ecrire un programme qui lit NMAX valeurs entières dans un tableau, qui effectue la saisie de chaque variable, qui calcule leur somme, stocke le résultat obtenu dans la variable result, affiche tous les nombres ainsi que le résultat**
4. **Reprendre le programme précédent, ajouter le calcul de la moyenne, puis afficher les éléments du tableau case par case en indiquant leur indice, la valeur contenue et un message si la valeur est plus grande ou égale à la moyenne**
5. **Déclarer un tableau de dimension 2, de trois lignes et 4 colonnes, initialiser les cases du tableau avec des valeurs allant de 12 à 23, la première ligne contiendra les valeurs 12, 13, 14, 15 ; la deuxième 16, 17, 18, 19 ; la troisième 20, 21, 22, 23 ; afficher les cases du tableau en utilisant la notation indicée**
6. **Reprendre l'exercice précédent avec la notation en pointeur pour la ligne**
7. **Reprendre l'exercice avec une notation en pointeur uniquement**
8. **Déclarer trois matrices d'entiers de taille 3x3, saisir les coefficients de deux matrices et additionner les deux matrices en stockant le résultat dans la troisième**
9. **Reprendre l'exercice en multipliant les deux matrices**

Exercice 5:

Déclaration et usage des fonctions, passage de paramètres

1. **Reprendre le problème de l'échange de deux variables entières x et y en créant une fonction 'swap' de type void qui effectue le même travail. On utilisera obligatoirement 2 paramètres de type pointeurs sur entier.**
2. **Pourquoi dans la question 1 utilise-t-on des pointeurs ? Aurait-on pu utiliser des entiers ? Justifier.**
3. **Ecrire une fonction C qui effectue les tâches suivantes : demande d'un nombre entier plus petit ou égal à 10 à l'utilisateur, ce nombre représentera le nombre de cases n d'un tableau d'entiers de taille MAX=10 que l'on veut saisir, saisit des n premières cases du tableau. La fonction retournera comme résultat le nombre d'entiers lus. Le tableau manipulé sera transmis à l'aide d'un paramètre de type pointeur sur tableau d'entier. On supposera que l'utilisateur saisit un nombre n qui est compris entre 0 et 10. On détaillera le passage de paramètres de type tableaux.**
4. **Ecrire une fonction C ayant comme paramètre un pointeur sur un tableau d'entiers t et un nombre entier n qui recherche le minimum sur les n premières cases de t. On modifiera ensuite la fonction pour pouvoir**

rechercher le maximum. On comptera le nombre d'opérations réalisées par type (ex : on a fait 6 tests et 2 affectations).

5. **Soit le programme suivant : #include <stdio.h> int main(int argc, char * argv[]) {while(--argc>0) printf("%s",*++argv) ; printf("\n") ;}. Indenter ce programme correctement et étudier son fonctionnement, ainsi que le passage de paramètre de la fonction principale.**

Exercice 6 :

Exercices supplémentaires sur les pointeurs. On suppose qu'un int occupe 4 octets en mémoire.

1. **Dites en quoi les lignes de code suivantes sont erronées.**

`int a , b ;`
`b = 3 ;`
`&a = b ;`
	`int *p , a`
	`a = 4 ;`
	`p = a ;`
	`*p = *p + 1 ;`
		`int *p , a ;`
		`a = 2 ;`
		`*p = a ;`

2. **Examinez les valeurs que prennent a, p et *p à chaque ligne de ce code.**

```
int  *p , a ;/* on suppose que l'adresse de a est 2000 */
a = 10 ;
p = &a
a = a - 1 ;
*p = a + 1 ;
*p = *p + 1 ;
p = p + 1 ;
```

3. **Soit le code C suivant :**

```
int  *p1, *p2 , tab[2] ;/* on suppose que l'adresse de tab[0] est 2000 */
tab[0] = 3 ;
tab[1] = 5 ;
p1 = tab ;
p2 = p1 + 1 ;
```

Après exécution, que valent tab, p1, *p1, p2, *p2 ?

4. **Soit le programme C suivant :**

```
#define      TAILLE 3

void truc (a , b)
int *a , b ;
{
      int i ;
      for (i = 0 ; i < b ; i++) *(a+i) = a[i] + 1 ;
}

main() {
      int tab[TAILLE] , i , *p ;
      p = &tab[0] ;
   for (i = 0 ; i < TAILLE ; i++)   *(p + i) = i ;
      truc(tab , TAILLE) ;
}
```

- **Sous quelle autre forme peut-on exprimer &tab[0] ?**
- **A quoi correspond *(tab + i) ?**
- **Exécutez ce programme à la main**
- **Le tableau tab aura-t-il été modifié au retour de la fonction tab ?**
- **Que fait ce programme ?**