TD1/2/3 - Rappels de C

Exercice 1 - Introduction (tous)

1. Écrire le programme C le plus court possible.
2. En utilisant la directive préprocesseur *define*, créer l'identificateur *PI* et lui associer la valeur *3.14*.
3. Ajouter la déclaration d'une variable entière dont l'identificateur sera *rayon*.
4. Initialiser la variable *rayon* avec la valeur 10, puis calculer la surface du cercle associé en stockant le résultat dans une variable dont l'identificateur sera *surface*.
5. Demander de saisir deux nombres entiers x et y et les afficher. Inverser leurs valeurs et les afficher de nouveau.
6. Déclarer une variable de type caractère et l'initialiser avec le caractère '7'. Affecter ensuite la valeur numérique 98 à cette variable. Afficher la valeur de la variable après l'initialisation et après l'affectation.

Exercice 2 - Entrées/Sorties (tous)

1. Écrire sur la sortie standard la chaîne de caractères "Bonjour !".
2. Ajouter un saut de ligne avant et après "Bonjour !".
3. Afficher "entrez un caractère : " et laisser l'utilisateur saisir un caractère. Sauter deux lignes et afficher sur une ligne "signe : *<le caractère>*, code asccii décimal : *<le code ascii décimal>*, code ascii hexadécimal : *<le code ascii hexadécimal>*".
4. Lire un entier et l'afficher en décimal, en octal et en hexadécimal.
5. Lire deux réels et les afficher. Calculer la somme et l'afficher.

Exercice 3 - Structures conditionnelles (1 à 5)

1. Afficher un message si un nombre entier saisi est compris entre 0 et 10.
2. Afficher un message si un nombre entier saisi est compris entre 7 et 20 ou si il vaut 127.
3. Afficher un message si un nombre entier saisi n'est pas nul.
4. Afficher un message si un caractère saisi est un chiffre et un message différent si c'est une lettre de l'alphabet (minuscule ou majuscule).
5. Lire un caractère (*getchar()*). Afficher "un" si c'est le caractère '1', "deux" si c'est '2', "trois" si c'est '3' et "autre caractère" sinon.
6. Lire une année de naissance. Calculer l'âge et afficher "Pour l'instant ça va" si celui-ci est plus petit ou égal à 50 ans et "Vous êtes sur la mauvaise pente" si il dépasse 50 ans.
7. Ajouter un test logique qui vérifie que l'âge est compris entre 0 et 130 inclus. Si c'est le cas alors effecuer le traitement de l'exercice précédent. Sinon afficher le message "age saisi invalide".
8. Lire un caractère. Afficher "c'est un bon caractère" si ce caractère est 'n', 'N', 'f', 'F', '1' ou '6'.
9. Vérifier si un nombre entier saisi est pair ou impair.

Exercice 4 - Expressions, opérateurs et conversions (1 à 8)

1. Initialiser une variable de type réel avec l'expression 5/2 et afficher sa valeur. Étudier les conversions automatiques de type.

2. Déclarer les variables entières a, b, c. Demander à l'utilisateur de saisir les valeurs et les afficher.
3. Ajouter le calcul de la somme des valeurs dans une variable *somme* et l'afficher.
4. Ajouter le calcul de la moyenne dans une variable *moyenne* et l'afficher.
5. Ajouter le calcul du produit des nombres dans une variable *produit* et l'afficher.
6. Calculer et afficher le nombre de caractères compris entre le 'a' et le 'A' dans la table des codes ASCII
7. Demander de saisir un caractère. Le transformer en majuscule si c'est une lettre de l'alphabet minuscule (le laisser inchangé sinon) et l'afficher.
8. Déclarer les variables entières a, b et c. Affecter à c la valeur 7, incrémenter c, affecter à a la valeur de c, incrémenter c, décrémenter c, affecter à b la valeur de c, décrémenter b, affecter à a la valeur de b. Que valent a, b et c ?
9. Lire un nombre entier et indiquer si il est divisible par 7.
10. Lire un nombre entier représentant une température exprimée en degrés Fahrenheit. Convertir en degrés centigrade et afficher les deux températures. La formule mathématique de conversion est : $\frac{5}{9}(temp\_farh - 32)$.
11. Ajouter un affichage avec 3 chiffres avant la virgule et 1 chiffre après la virgule.
12. Tester si une année est bissextile. On rappelle qu'il y a aussi une année bissextile tous les 400 ans.

Exercice 5 - Structures itératives (1 à 5)

1. Afficher les nombres de 0 à 9 avec une boucle *for*, puis une boucle *do... while*, puis une boucle *while*.
2. Afficher la table des codes ASCII pour les caractères au delà de la valeur 31 et jusqu'à la valeur 127.
3. Afficher les caractères compris entre 'A' et 'Z' puis entre 'a' et 'z'. Afficher le caractère, le code ASCII en décimal et le code ASCII en hexadécimal.
4. À l'aide d'une instruction *switch* et de la fonction *getchar()* : demander à l'utilisateur de saisir un caractère. Si le caractère est compris dans l'ensemble ('o', 'O', 'n', 'N', '0', '1', '2', '3'), afficher "c'est un bon caractère", sinon afficher "c'est un mauvais caractère ". Réitérer la demande jusqu'à ce que l'utilisateur saisisse le caractère 'n' ou 'N'.
5. Calculer la fonction factorielle définie par $F_0 = 1$; $F_n = n * F_{n-1}$.
6. Calculer la suite définie par $S_0 = 1$; $S_n = S_{n-1} + 7$.
7. Lire des caractères et les afficher jusqu'à ce que l'utilisateur saisisse le caractère `'\n'`. Afficher le nombre de caractères lus.
8. Calculer les intérêts cumulés à partir d'une somme s, sur n périodes de temps et avec un taux d'intérêt de t.
9. Lire une suite d'entiers strictement positifs et s'arrêter lorsque l'utilisateur rentre la valeur -1. Calculer la moyenne des valeurs saisies et l'afficher.
10. Afficher "Quelle table de multiplication voulez-vous afficher (tapez 0 pour sortir) ?". Laisser l'utilisateur saisir un caractère. Si le caractère est compris entre '1' et '9' alors afficher la table correspondante puis réitérer le processus. Sinon afficher "ce n'est pas dans les possibilités du programme, recommencez !" et réitérer le processus.
11. Modifier le programme précédent pour travailler en base 16 (hexadécimal). On rappelle que les digits licites en base 16 sont ('0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C' , 'D', 'E', 'F'). Veiller à traiter convenablement le cas des caractères.

Exercice 6 - Pointeurs (tous)

1. Déclarer une variable i de type entier et l'initialiser à 5. Afficher l'adresse de i et la valeur de i.
2. Ajouter une variable j de type pointeur vers un entier et l'initialiser avec l'adresse de i. Afficher l'adresse de j, la valeur de j et la valeur pointée par j.
3. Incrémenter la donnée pointée par j. Afficher la valeur de i et la valeur pointée par j.
4. Multiplier i par 5. Afficher la valeur de i et la valeur pointée par j.
5. Incrémenter j. Afficher la valeur de i, la valeur de j et la valeur pointée par j.

Exercice 7 - Tableaux et chaînes de caractères (1 à 7)

1. Déclarer un tableau de 10 entiers. À l'aide d'une boucle, affecter la valeur 7 à tous les indices du tableau.
2. Déclarer un tableau de 5 entiers en l'initialisant avec les valeurs décroissantes allant de 4 à 0.
3. Déclarer une chaîne de caractères en l'initialisant à "bonjour". Afficher tous les caractères de cette chaîne jusqu'au caractère `\0`, puis afficher la chaîne complète.
4. Demander à l'utilisateur de saisir une chaîne de caractères. Remplacer le premier caractère 'o' par le caractère 'a' et le premier caractère 'n' par le caractère 'd' puis afficher la chaîne complète.
5. Déclarer un tableau d'entiers de taille *NMAX*. Demander la saisie de chaque valeur du tableau, calculer la somme des valeurs, stocker le résultat dans une variable *somme* et l'afficher.
6. Reprendre le programme précédent. Ajouter le calcul de la moyenne puis afficher les éléments du tableau case par case en indiquant l'indice, la valeur et un message si celle-ci est plus grande ou égale à la moyenne.
7. Déclarer un tableau de dimension 2 (3 lignes et 4 colonnes) et l'initialiser avec les 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 indices et valeurs du tableau.
8. Reprendre l'exercice précédent avec la notation en pointeurs pour la ligne.
9. Reprendre l'exercice avec une notation en pointeurs uniquement.
10. 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.
11. Reprendre l'exercice en multipliant les deux matrices.
12. Déclarer un tableau de 10 chaînes de caractères de 15 caractères chacune, initialiser toutes les chaînes avec le caractère `\0`.
13. Ajouter la saisie des 10 chaînes au clavier, puis les afficher à l'écran.

Exercice 8 - Fonctions et passage de paramètres (1 à 9)

1. Reprendre le problème de l'échange de deux variables entières x et y en créant une fonction *void swap(int * a, int * b)*. Justifier l'utilisation de pointeurs.
2. Écrire une fonction *my_ isascii(char c)* qui retourne 1 si c fait partie de la table ASCII et 0 sinon.
3. Écrire une fonction *my_ strlen* qui retourne la longueur d'une chaîne de caractères.
4. Écrire une fonction qui demande de saisir un nombre entier n plus petit ou égal à 10 à l'utilisateur. n est le nombre de cases qui seront utilisées dans un tableau d'entiers de taille $MAX = 10$ transmis en paramètre de la fonction. Demander ensuite de saisir les n valeurs du tableau. La fonction retournera n.
5. Écrire une fonction *min* qui renvoie la plus petite valeur comprise dans un tableau d'entiers. Quels sont les paramètres nécessaires à cette fonction ?
6. Écrire une fonction *en_ domain(int a, int b, int c)* qui retourne 1 si $a \in [b; c]$ et 0 sinon.
7. Écrire une fonction qui transforme tous les caractères minuscules d'une chaîne en caractères majuscules.
8. Soit le programme suivant :

```
include <stdio.h>
int main(int argc, char * argv[]) {
  while(--argc>0) {
    printf("%s",*++argv);
    printf("\n") ;
  }
}
```

   Étudier son fonctionnement, ainsi que le passage de paramètres de la fonction principale.
9. Reprendre le calcul de la factorielle sous la forme d'une fonction récursive.
10. Écrire une fonction *my_ isalpha(char c)* qui retourne 1 si c est une lettre.
11. Écrire une fonction *my_ isdigit(char c)* qui retourne 1 si c est un chiffre
12. Écrire une fonction *somme* (2 paramètres) qui calcule et retourne la somme de deux entiers.
13. Reprendre la fonction *somme* et écrire la fonction *somme2* qui retourne son résultat dans un argument d'appel supplémentaire (donc la fonction ne retourne plus aucune valeur).
14. Écrire une fonction *fact1* (1 paramètre) qui calcule la factorielle de façon itérative.

15. Écrire une fonction *my_ toupper* (1 paramètre) qui transforme un caractère c en la valeur de la majuscule associée (s'il y en a une, sinon ne modifie rien).
16. Reprendre l'exercice précédent et écrire *my_ tolower* (1 paramètre) qui effectue le travail inverse.
17. Écrire la fonction *my_ atoi* (1 paramètre) qui prend une chaîne de caractères en paramètre et retourne l'entier lu depuis celle-ci.
18. Écrire la fonction *my_ ltoa* (2 paramètres) qui transforme un entier *long* en une chaîne de caractères.
19. Écrire une fonction *my_ strcmp* (2 paramètres) qui renvoie vrai si deux chaînes de caractères sont égales, faux sinon.
20. Écrire une fonction *my_ strcat* (2 paramètres) qui concatène une chaîne source à une chaine destination et retourne un pointeur sur le résultat.
21. Écrire une fonction *my_ strcpy* (2 paramètres) qui recopie une chaîne source dans une chaine destination et retourne un pointeur sur le résultat.
22. Écrire une fonction *my_ strnset* (3 paramètres) qui recopie *nbr* fois le caractère *c* dans une chaîne destination et retourne un pointeur sur le résultat.
23. Écrire une fonction *my_ strdup* (1 paramètre) qui recopie une chaîne de caractères vers un emplacement réservé par *malloc* et retourne un pointeur sur le résultat.

Exercice 9 - Structures (1 à 7)

1. Détailler la déclaration d'une structure ainsi que le mode d'accès direct et en utilisant des pointeurs.
2. Déclarer la structure *Un_ Tableau_ Entier* contenant un tableau d'entiers de taille $NMAX$ et une variable entière nommée *ncase* donnant le nombre de cases utilisées dans le tableau.
3. Déclarer la structure *menu* contenant un tableau (d'au plus 20 items) de chaîne de caractères (d'au plus 60 caractères). La structure contiendra aussi un entier n représentant le nombre d'items du menu.
4. Ajouter une fonction qui retourne la longueur de la plus longue chaîne parmi celles d'un menu.
5. Ajouter une fonction qui affiche les items d'un menu en les centrant par rapport à la plus longue chaîne. Chaque item est précédé de son numéro d'indice (entre 1 et n).
6. Ajouter une fonction qui affiche le menu et demande à l'utilisateur "Veuillez choisir votre menu (q ou Q pour quitter)". Si la saisie est valide, retourner l'indice du menu saisi (entre 1 et n). Si l'utilisateur tape 'q' ou 'Q', retourner 0. Si l'utilisateur tape autre chose, effacer l'écran et afficher de nouveau le menu.
7. Déclarer la structure *tonneau*. Un tonneau est 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 $V = \frac{\pi L}{4}(d + \frac{2}{3}(D - d))^2$. Enfin un tonneau contient un liquide (du vin, du vinaigre, de la bière ou de l'huile). Utiliser l'instruction *typedef*.
8. Déclarer une structure *Une_ Date*, avec le jour, le mois et l'année ; initialiser chaque variable de ce type avec la date du 01/01/01.
9. Déclarer une structure *Un_ Etudiant* décrivant un étudiant avec son nom, son prénom, sa date de naissance, son niveau (branche ou tronc commun) et enfin son numéro de semestre.
10. Écrire une fonction *Init_ Tableau*, qui affecte à toutes les cases d'un tableau de type Un_Tableau_Entier la valeur 0.
11. Écrire une fonction *Saisir_ Tableau_ Entier* qui demande à l'utilisateur le nombre de valeurs à saisir. Afficher un message d'erreur si la valeur est négative ou dépasse NMAX et réitérer la demande. Sinon, effectuer la saisie des entiers à condition que ceux-ci soient strictement positifs.
12. Écrire une fonction qui retourne la moyenne d'un tableau de type *Un_ Tableau_ Entier*.
13. Écrire une fonction qui effectue la saisie d'une variable de type *Une_ Date*.
14. Écrire une fonction qui effectue la saisie une variable de type *Un_ Etudiant*.
15. Déclarer le type *Un_ Tableau_ Etudiant* en composant les types ci-dessus.
16. Écrire une fonction qui réalise la saisie des étudiants de NF16 .

Exercice 10 - Plus de pointeurs (tous)

On suppose qu'un *int* occupe 4 octets en mémoire.

1. Trouver l'erreur dans les 3 codes suivants.

```
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. Déterminer les valeurs de a, p, $*p$ à chaque ligne du code suivant :

```
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. Déterminer les valeurs de tab, $p1$, $*p1$, $p2$, $*p2$ après exécution du code 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;
```

4. Soit le programme suivant :

```
#define TAILLE 3
void truc (int * a, int b)
{
    int i ;
    for (i=0; i<b; i++)
      *(a+i) = a[i] + 1;
}
int main() {
    int tab[TAILLE], i, *p ;
    p = &tab[0] ;
    for (i = 0 ; i < TAILLE ; i++)
      truc(tab, TAILLE);
    return 0;
}
```

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