NF16 Exercices d'entraînement **UTC**

Partie 1 : (tous)

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

Partie 2 : (tous)

printf et scanf, directive #include, formatage d'entrée/sortie, caractères spéciaux

1. Ecrire un programme C qui écrit la chaîne de caractères "Bonjour !"
2. Reprendre le programme précédent et faire sauter une ligne blanche avant et après "Bonjour !"
3. Ecrire un programme C qui affiche "entrez un caractère : ", lit un caractère, saute deux lignes blanches, affiche sur une ligne "signe : ", le caractère, " code asccii décimal : ", le code ascii décimal, " code ascii hexadécimal : ", et enfin le code ascii hexadécimal
4. Ecrire un programme C qui lit un entier et l'affiche en décimal, en octal et en hexadécimal
5. Ecrire un programme C qui lit deux réels, les affiche, calcule leur somme et affiche le résultat

Partie 3 : (tous)

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.

Partie 4 : (1 à 5)

Opérateurs de comparaison, opérateurs logiques, convention logique du C, instruction if et switch, instruction composée, instruction break

1. Ecrire un programme C qui teste si un nombre entier saisi est compris entre 0 et 10

2. Construire un test logique pour vérifier qu'un entier est compris entre 7 et 20 ou qu'il vaut 127
3. Construire un test logique qui vérifie qu'un entier n'est pas nul
4. Ecrire un programme C qui teste si un caractère saisi au clavier est un chiffre ou si c'est un caractère compris entre 'a' et 'z' ou encore entre 'A' et Z'
5. Ecrire un programme en utilisant l'instruction switch qui lit un caractère (fonction getchar()), qui affiche "un" si le caractère '1' a été saisi, "deux" si c'est le caractère '2', "trois" si c'est '3' et "autre caractère" sinon
6. Ecrire un programme C qui lit une année de naissance, calcule l'âge et affiche "Pour l'instant cela va" si l'âge est plus petit ou égal à 50 ans et affiche "Vous êtes sur la mauvaise pente" si l'âge dépasse 50 ans
7. Le programme précédent souffre d'un problème majeur, en effet on peut saisir une valeur négative (ce qui est absurde), contruisez un test logique qui vérifie que l'âge est compris entre 0 et 130 inclus, si c'est le cas alors on effectuera le traitement de l'exercice précédent sinon on affichera un message d'erreur
8. Ecrire un programme qui effectue la saisie d'un caractère, qui affiche "c'est un bon caractère" si ce caractère est 'n' ou 'N' ou 'f' ou 'F' ou '1' ou '6' ou ' ' ou 'a' ou 'A' ou 'u' ou 'U' ou 't' ou 'T' ou 'o' ou 'O' ou 'm' ou 'M' ou 'n' ou 'N' ou 'e' ou 'E'
9. Ecrire un programme qui lit un entier et affiche à l'écran si c'est 0, un nombre pair ou un nombre impair

Partie 5 : (1 à 7)

Opérateurs arithmétiques binaires et unaires, opérateurs d'affectation, conversions

1. Ecrire un programme comportant les variables entières a, b, c, d, e, f, g, h, i, j ; 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
2. Ecrire un programme qui calcule la valeur mathématique 5/2, qui l'affecte à une variable de type réel (type float) et qui affiche le résultat. Etudier les conversions automatiques de type.
3. Reprendre le programme de la somme de variables, calculer la moyenne, stocker le résultat dans la variable moyen, et afficher le résultat
4. Reprendre le programme précédent calculer le produit des nombres, stocker le résultat dans la variable prod, et afficher le résultat
5. Ecrire un programme qui calcule et affiche le nombre de caractères compris entre le 'a' et le 'A' dans la table des codes ASCII
6. Modifier le programme précédent de façon à effectuer la saisie d'un caractère et à le transformer en majuscule si c'est un caractère compris entre 'a' et 'z' et à le laisser inchangé sinon
7. Ecrire un programme qui déclare les variables entières a, b et c ; qui affecte à c la valeur 7, qui l'incrémente, qui affecte a avec la valeur de c puis incrémente c, qui décrémente c puis affecte la valeur à b puis qui décrémente b et affecte la valeur à la variable a, que valent respectivement a, b et c ?
8. Ecrire un programme qui vérifie qu'un nombre entier f est divisible par 7
9. Ecrire un programme qui lit un mombre entier représentant une température exprimée en degré Fahrenheit, qui la convertit en degré centigrade, la formule mathématique est : 5/9(temp_farh-32) et qui affiche les deux températures
10. Reprendre le programme précédent et réaliser un affichage avec 3 chiffres avant la virgule et 1 chiffre après la virgule
11. Ecrire un programme qui teste si une année est bissextile, on rappelle qu'il y a aussi une année bissextile tous les 400 ans

Partie 6 : (1 à 5)

les boucles : intructions while, do ... while, for

1. Ecrire un programme en C pour afficher les nombres de 0 à 9 avec une boucle for, puis une boucle do, puis une boucle while
2. Ecrire un programme qui affiche la table des codes ASCII pour les caractères au delà de la valeur 31 et jusqu'à la valeur 127 (incluse)
3. Ecrire un programme en C pour afficher les caractères compris entre [‘A’ ...‘Z’] et entre [‘a’ ...’z’], on affichera le caractère, le code ASCII en décimal, le code ASCII en hexadécimal
4. Ecrire un programme (avec un switch) qui affiche "entrez un caractere : ", qui lit un caractère, si ce caractère est compris dans l'ensemble ('o', 'O', 'n', 'N', '0', '1', '2', '3') alors on affichera "c'est un bon caractère", sinon on affichera "c'est un mauvais caractère " et on réitérera le processus sauf s'il s'agit du caractère 'n' ou 'N'
5. Ecrire un programme qui calcule la fonction factorielle définie par F0 = 1; Fn = n*Fn-1
6. Ecrire un programme qui calcule la suite définie par S0 = 1; Sn = Sn-1 + 7
7. Ecrire un programme qui lit des caractères au clavier et qui s'arrête quand on tape le caractère '\n'
8. Reprendre le programme précédent, afficher les caractères lus en écho et compter les caractères
9. Ecrire un programme qui calcule les intérêts cumulés à partir d'une somme s, sur n périodes de temps et avec un taux d'intérêt de t
10. Ecrire un programme qui lit une suite d'entiers strictement positifs et qui s'arrête si on rentre la valeur -1
11. Modifier le programme précédent en ajoutant le calcul de la moyenne
12. Ecrire un programme qui demande "Quelle table de multiplication voulez-vous, tapez 0 (zero) pour sortir ?", qui saisit un caractère, si le caractère est compris entre '1' et '9' alors on fera afficher au programme la table correspondante puis on réitérera le processus, sinon on affichera "ce n'est pas dans les possibilités du programme, recommencez !" et on réitérera le processus
13. Reprendre le programme précédent et 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'), on veillera à traiter convenablement le cas des caractères minuscules

Partie 7 : (1 à 8)

Déclaration de variables structurées de type tableau, convention pour les chaînes de caractères, initialisation de tableau à la déclaration, fonction d'entrée/sortie sur les chaînes de caractères, 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. Déclarer et initialiser la chaîne de caractères "bonjour"
4. Faire afficher tous les caractères de cette chaîne jusqu'au caractère '\0', puis afficher la chaîne complète
5. Ecrire les instructions qui remplacent le premier caractère 'o' par le caractère 'a' et le caractère 'n' par le caractère 'd' puis afficher la chaîne complète
6. 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
7. 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

8. 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
9. Reprendre l'exercice précédent avec la notation en pointeur pour la ligne
10. Reprendre l'exercice avec une notation en pointeur uniquement
11. 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
12. Reprendre l'exercice en multipliant les deux matrices
13. Déclarer un tableau de 10 chaînes de caractères de 15 caractères chacune, initialiser toutes les chaînes avec le caractère '\0'
14. Augmenter le programme précédent en effectuant la saisie des 10 chaînes au clavier, puis en les affichant à l'écran

Partie 8 : (1 à 10)

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 la fonction my_isascii (1 paramètre) qui retourne une valeur non nulle si un caractère est compris entre 0 et 127
4. Ecrire une fonction my_strlen ayant comme paramètre un pointeur sur une chaîne de caractères et retournant la longueur de la chaîne. On rappellera la déclaration et l'usage des chaînes de caractères.
5. 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.
6. 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).
7. Ecrire une fonction C qui retourne 1 si un paramètre a entier est entre un paramètre b entier et un paramètre c entier (bornes comprises) et 0 sinon. Elle s'appellera en_domaine. Ex: en_domaine(1,0,5) retourne 1
8. Ecrire une fonction C qui transforme tous les caractères minuscules d'une chaîne en caractères majuscules
9. 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 sont fonctionnement, ainsi que le passage de paramètre de la fonction principale.
10. Reprendre le calcul de la factorielle sous la forme d'une fonction récursive.
11. Ecrire la fonction my_isalpha (1 paramètre) qui retourne une valeur non nulle si un caractère est une lettre
12. Ecrire la fonction my_isdigit (1 paramètre) qui retourne une valeur non nulle si un caractère est un chiffre
13. Ecrire une fonction somme (2 paramètres) qui calcule et retourne la somme de deux entiers

14. 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)
15. Ecrire la fonction fact1 (1 paramètre) qui calcule la factorielle de façon itérative
16. Ecrire la 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)
17. Reprendre l'exercice précédent et écrire my_tolower (1 paramètre) qui effectue le travail inverse
18. Ecrire la fonction my_atoi (1 paramètre) qui transforme une chaîne de caractères en un entier
19. Ecrire la fonction my_ltoa (2 paramètres) qui transforme un entier long en une chaîne de caractères
20. Ecrire une fonction my_swap (2 paramètres) qui échange le contenu de deux entiers
21. Ecrire une fonction my_strcmp (2 paramètres) qui répond vrai si deux chaînes de caractères sont égales, faux sinon
22. Ecrire une fonction my_strcat (2 paramètres) qui recopie une chaîne source à la fin d'une chaine destination et retourne un pointeur sur le résultat
23. Ecrire une fonction my_strcpy (2 paramètres) qui recopie une chaîne source dans une chaine destination et retourne un pointeur sur le résultat
24. Ecrire une fonction my_strnset (3 paramètres) qui recopie nbr fois le caractère c dans une chaîne destination et retourne un pointeur sur le résultat
25. Ecrire une fonction my_strdup (1 paramètre) qui fait un double d'une chaîne dans un emplacement réservé par malloc et retourne un pointeur sur le résultat

Partie 9 : (1 à 7)

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 une structure menu comme étant 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. Ecrire une fonction C qui retourne la longueur de la plus longue chaîne parmi celles d'un menu
5. Ecrire une fonction C qui affiche un menu en le centrant par rapport à sa plus grande chaîne, chaque item devant être précédé de son numéro d'indice, tous les items étant affichés, on sautera 2 lignes puis on affichera "Q pour quitter"
6. Ecrire une fonction C qui affiche un menu (de n items), puis effectue une saisie, si la chaîne saisie ne correspond pas à une des possibilités du menu (entre 1 et n) alors on effacera l'écran et ré-affichera le menu, sinon on retournera un entier correspondant à l'item sélectionné et 0 si l'utilisateur tape 'q' ou 'Q'
7. 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 : $V = 3.14159\ L\ (d/2+2/3(D/2-d/2))^2$. Enfin un tonneau contient quelque chose (du vin, du vinaigre, de la bière ou de l'huile). On utilisera la syntaxe avec **typedef**.
8. Ecrire la déclaration d'une structure qu'on appellera Une_Date, avec le jour, le mois et l'année ; on initialisera chaque variable de ce type avec la date du 01/01/01

9. Ecrire la déclaration d'une structure qu'on appellera 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. Ecrire une fonction Init_Tableau, qui initialise toutes les cases d'un tableau de type Un_Tableau_Entier avec la valeur 0
11. Ecrire une fonction Czi_Tableau_Entier qui demande à l'utilisateur le nombre de valeurs à saisir, qui affiche un message d'erreur si on entre une valeur négative ou si on dépasse NMAX puis réitère la demande ; sinon effectue la saisie des entiers à condition que ceux-ci soient strictement positifs
12. Ecrire une fonction qui retourne la moyenne d'un tableau de type Un_Tableau_Entier
13. Ecrire une fonction qui effectue la saisie d'une variable de type Une_Date
14. Ecrire une fonction qui saisit une variable de type Un_Etudiant
15. Déclarer un type Un_Tableau_Etudiant en composant les types ci-dessus
16. Ecrire une fonction qui réalise la saisie des étudiants de NF16

Partie 10 : (tous)

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 ?