TP N°1

Tout premier programme C

Exercice 1 : Syntaxe

Indenter correctement le programme suivant :

```
#include <stdlib.h> #include
<stdio.h> void main (
) { int x ; printf ( "Entrer la valeur de x\n" ) ;
scanf ( "%d", &x ) ; if ( x < 0) printf (
"x est un entier negatif \n" )
; else printf ( "x est un entier
positif \n" ) ; }
```

Exercice 2 : Types de données

1. Expliquer la différence entre ces deux objets du langage C : + et '+'
2. Quels sont les types des données suivantes ? Donner également leurs valeurs.

1	'1'	"1"	1.0	.1e1

3. Pour imprimer un caractère à l'aide de sa valeur, disons c, dans le code ASCII il suffit d'utiliser l'instruction printf ("%c",c). Écrire un programme dans lequel on déclare et initialise un caractère, et qui affiche le caractère suivant (par exemple, si on initialise le caractère à 'd', le programme affiche e).

Exercice 3 : Expressions

Éliminer les parenthèses superflues dans les expressions suivantes :

```
a = ( x + 5)            /*expression 1 */
a = ( x = y ) + 2       /*expression 2 */
a = ( x == y )          /*expression 3 */
( i++) * (n + p)        /*expression 4 */
```

Exercice 4 : E/S simple

Écrire un programme qui demande à l'utilisateur de saisir un entier et qui affiche à l'écran le caractère correspondant.

Exercice 5 :E/S conversationnelles

Quels résultats fournit le programme suivant :

```
#include <stdio.h>
#include <stdlib h>
void main ( ) {
    int i , j , n ;
    i = 0 ; n = i++;
    printf ( "A : i=%d n=%d\n" , i , n ) ;
    i = 10 ; n = ++i ;
    printf ( "B : i=%d n=%d\n" , i , n ) ;
    i = 20 ; j = 5 ; n = i++ * ++j ;
    printf ( "C : i=%d j=%d n=%d\n" , i , j , n ) ;
    i = 15 ; n = i += 3 ;
    printf ( "D : i=%d n=%d\n" , i , n ) ;
```

```
        i = 3 ; j = 5 ; n = i *= --j ;
        printf ( "E : i=%d j=%d n=%d\n" , i , j , n ) ;
}
```

Exercice 6 : E/S conversationnelles

Quels résultats fournit le programme suivant :

```
#include <stdio.h>
#include <stdlib.h>
void main ( ) {
        int n=543;
        int p=5;
        float x=34.5678;
        printf ( "A : %d %f \n" ,n , x ) ;
        printf ( "B : %4d %10f \n" ,n , x ) ;
        printf ( "C : %2d %3f \n" ,n , x ) ;
        printf ( "D : %10.3 f  %10.3 e \n" ,x , x ) ;
        printf ( "E : %-5d  %f \n" ,n , x ) ;
        printf ( "F : %*d\n" ,p , n ) ;
        printf ( "G : %*.*f \n" , 1 2 , 5 , x ) ;
        printf ( "H : %x : %8x : \n" ,n , n ) ;
        printf ( " I : %o : %8o : \n" ,n , n ) ;
}
```

Exercice 7 : E/S conversationnelles

1) Quelles seront les valeurs des deux variables n et p (de type int), par l'instruction suivante :

```
        scanf ( "%d %d",&n, &p ) ;
```

Lorsqu'on lui fournit les données suivantes (le symbole ^ représente un espace et le symbole @ représente une fin de ligne, c'est-à-dire une "validation").

(a) 253^45@
(b) ^253^@
^^4^5@

2) Quelles seront les valeurs des deux variables n et p (de type int), par l'instruction suivante :

```
        scanf ( "%4d %2d",&n,&p ) ;
```

Lorsqu'on lui fournit les données suivantes :

(a) 12^45@
(b) 123456@
(c) 123456^7@
(d) 1^458@
(e) ^^^4567^^8912@

TP N°2

Les structures conditionnelles

Exercice 1:

Réaliser en C un algorithme qui lit trois valeurs entières (A, B et C) au clavier et qui affiche la plus grande des trois valeurs, en utilisant:

a) si-sinon et une variable d'aide MAX
b) si-sinon-si sans variable d'aide
c) les opérateurs conditionnels et une variable d'aide MAX
d) les opérateurs conditionnels sans variable d'aide

Exercice 2:

Réaliser en C un algorithme qui lit trois valeurs entières (A, B et C) au clavier. Trier les valeurs A, B et C par échanges successifs de manière à obtenir (A <= B <= C) puis afficher les trois valeurs.

Exercice 3:

Réaliser en C un algorithme qui lit deux valeurs entières (A et B) au clavier et qui affiche :

e) le signe du produit de A et B sans faire la multiplication.
f) le signe de la somme de A et B sans faire l'addition.

Exercice 4:

Réaliser en C un algorithme qui permet de saisir un numéro de couleur de l'arc-en-ciel et d'afficher la couleur correspondante :

1: rouge, 2 : orangé, 3 : jaune, 4 : vert, 5 : bleu, 6 : indigo et 7 : violet.

Exercice 5:

On veut déterminer si une année A est bissextile ou non.
Si A n'est pas divisible par 4 l'année n'est pas bissextile.
Si A est divisible par 4, l'année est bissextile sauf si A est divisible par 100 et pas par 400.

Exemples :

1980 et 1996 sont bissextiles car elles sont divisibles par 4
2000 est une année bissextile car elle est divisible par 400
2100 et 3000 ne sont pas bissextiles car elles ne sont pas divisibles par 400.

Réaliser en C un algorithme qui effectue la saisie de la donnée, détermine si l'année est bissextile ou non et affiche le résultat.

Exercice 6:

Un temps donné est représenté sous la forme: heure, minute et seconde de type naturel. On veut lui ajouter une seconde et afficher avec la même représentation le temps ainsi obtenu.

Réaliser en C un algorithme qui effectue la saisie des données, réalise le calcul et affiche les résultats.

TP N°3

Les structures itératives

Exercice 1:

Réaliser en C un algorithme qui permet de déterminer la somme des chiffres d'un nombre entier donné (exemple : pour N= 25418, on aura 2+5+4+1+8 = 20).

Exercice 2:

Un nombre réel X et un nombre entier N étant donnés, proposer un programme C qui fait calculer X^N. Étudier tous les cas possibles (N positive ou négative).

Exercice 3:

Les nombres de Fibonacci sont donnés par la récurrence :

$F_n = F_{n-2} + F_{n-1}$ avec $F_0 = 1$ et $F_1 = 1$.

Écrire un programme C qui affiche les 20 premiers nombres de Fibonacci.

Exercice 4:

Un entier naturel de trois chiffres est dit cubique s'il est égal à la somme des cubes de ses trois chiffres.

Exemple : 153 est cubique car $153 = 1^3 + 5^3 + 3^3$

Réaliser en C un algorithme qui cherche et affiche tous les entiers cubiques de trois chiffres.

Exercice 5:

Écrire un programme C qui permet de lire un entier positif et déterminer tous ses facteurs premiers.

Exemples:

30=2 * 3 * 5
36=2 * 2 * 3 * 3
99=3 * 3 * 11

Exercice 6:

Deux nombres entiers sont premiers entre eux s'ils n'ont pas d'autres diviseurs communs que 1.

- 7 et 13 n'ont que 1 comme diviseur commun donc 7 et 13 sont premiers entre eux.
- 12 et 32 ont plusieurs diviseurs communs : 1 ; 2 et 4 donc 12 et 32 ne sont pas premiers entre eux.

Écrire un programme C qui saisit deux entiers N1 et N2, vérifie et affiche s'ils sont premiers entre eux ou non.

Exercice 7:

Un nombre est dit palindrome s'il est écrit de la même manière de gauche à droite ou de droite à gauche.

Exemples : 101 ; 22 ; 3663 ; 10801, etc.

Écrire un programme C permettant de déterminer et d'afficher tous les nombres palindromes compris dans l'intervalle [100 .. 9999].

Exercice 8:

Réaliser en C un algorithme qui imprime pour un entier naturel n donné :

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
.....
.....
.....
1 2 3 4 5 6 ... n

Exercice 9 :

Deux entiers N1 et N2 sont dits frères si chaque chiffre de N1 apparaît au moins une fois dans N2 et inversement.

Exemples :

- Si N1 = 1164 et N2 = 614 alors le programme affichera : N1 et N2 sont frères
- Si N1 = 405 et N2 = 554 alors le programme affichera : N1 et N2 ne sont pas frères

Écrire un programme C qui saisit deux entiers N1 et N2, vérifie et affiche s'ils sont frères ou non.

Exercice 10:

On veut réaliser en même temps, à l'aide de soustractions successives, la division entière et le calcul du modulo de deux nombres naturels non nuls dividende et diviseur.

Réaliser en C l'algorithme qui effectue la saisie des données, réalise le calcul et affiche les résultats.

Exercice 11:

Réaliser en C un algorithme qui affiche la suite de tous les nombres parfaits inférieurs ou égaux à un nombre naturel non nul donné noté n. Un nombre est dit parfait s'il est égal à la somme de ses diviseurs autre que lui-même.

Exemple: 28 = 1 + 2 + 4 + 7 + 14

Voici la liste des nombres parfaits inférieurs à 10000 : 6, 28, 496, 8128.

Exercice 12:

Dans une entreprise, le calcul des jours de congés payés s'effectue de la manière suivante :

- Si une personne est rentrée dans l'entreprise depuis moins d'un an, elle a le droit à deux jours de congés par mois de présence, sinon à 28 jours au moins.
- Si c'est un cadre, s'il est âgé d'au moins 35 ans et si son ancienneté est supérieure ou égale à 3 ans, il lui est accordé deux jours supplémentaires. Si ce cadre est âgé d'au moins 45 ans et si son ancienneté est supérieure ou égale à 5 ans, il lui est accordé 4 jours supplémentaires, en plus des 2 accordés pour plus de 35 ans.

Réaliser en C l'algorithme qui calcule le nombre de jours de congés à partir de l'âge, de l'ancienneté et de l'appartenance au collège cadre d'un employé. Afficher le résultat.

TP N°4

Les fonctions et les tableaux en C

Exercice 1:

Écrire en C un algorithme d'une fonction Triangle qui permet de vérifier si les 3 nombres a, b et c peuvent être les mesures des côtés d'un triangle rectangle.
Remarque: D'après le théorème de Pythagore, si a, b et c sont les mesures des côtés d'un rectangle, alors $a^2= b^2+c^2$ ou $b^2=a^2+c^2$ ou $c^2 =a^2+b^2$

Exercice 2 :

Ecrire un programme en C qui lit deux nombre naturel non nul *m et n* et qui détermine s'ils sont amis. Deux nombres entiers n et m sont qualifiés d'amis, si la somme des diviseurs de n est égale à m et la somme des diviseurs de m est égale à n (on ne compte pas comme diviseur le nombre lui-même et 1). Proposer une solution modulaire.

Exercice 3 :

Réaliser en C un algorithme d'une fonction qui recherche le premier nombre entier naturel dont le carré se termine par n fois le même chiffre.
Exemple : pour n = 2, le résultat est 10 car 100 se termine par 2 fois le même chiffre.

Exercice 4: Récursivité simple

Soit la suite numérique Un suivante : Si n = 0 alors $U_0 = 4$ sinon si n > 0 alors $Un =2*U_{n-1} + 9$
Écrire une fonction C qui calcul le terme Un.

Exercice 5: Récursivité croisée

Écrire deux fonctions C qui permettent de calculer les $n^{èmes}$ (n passé en argument) termes des suites entières Un et Vn définies ci-dessous.

$$\begin{cases} U_0=1 \\ Un=Vn_{-1}+1 \end{cases} \qquad \begin{cases} V_0=0 \\ Vn=2*Un_{-1} \end{cases}$$

Exercice 6: Fibonnaci

La suite de FIBONNACI est définie par :

Fib (1)= 1 ; Fib (2)= 2 ;
Fib (n)= Fib (n-1)+ Fib (n-2) n >2;

Pour pouvoir programmer cette suite de FIBONNACI, on adopte la décomposition fonctionnelle suivante :
1. **init** : prépare les conditions nécessaires et suffisantes pour calculer **fib(3)**.
Programmer en C le sous-programme **init**.
2. **suivant** : calcule le terme suivant. Programmer en C le sous-programme **suivant**.
3. **terme** : rend le terme courant. Programmer en C le sous-programme **terme**.
4. **fib** : calcule le nombre de FIBONNACI d'un rang donné. Programmer en C le sous-programme **fib**.
5. **est_ce_fibo** : permet de voir si un nombre donné est un nombre de FIBONNACI.
Programmer en C le sous-programme **est_ce_fibo**.

Exercice 7 : Somme, produit et moyenne des éléments

Écrire un programme C qui lit la dimension N (5<N<20) d'un tableau T de type int. Remplit ce tableau par des valeurs entrées au clavier et affiche le tableau.
Calculer et afficher ensuite la somme, le produit et la moyenne des éléments du tableau.

Exercice 8 : Occurrence de 0

Écrire un programme C qui lit la dimension N d'un tableau T de type int (dimension maximale: 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau.
Effacer ensuite toutes les occurrences de la valeur 0 dans le tableau T et tasser les éléments restants. Afficher le tableau résultant.

Exercice 9 : Inverse

Écrire un programme C qui lit la dimension N d'un tableau T de type int (dimension maximale: 50 composantes), remplit le tableau par des valeurs entrées au clavier et affiche le tableau. Ranger ensuite les éléments du tableau T dans l'ordre inverse sans utiliser de tableau d'aide. Afficher le tableau résultant.
Idée: Echanger les éléments du tableau à l'aide de deux indices qui parcourent le tableau en commençant respectivement au début et à la fin du tableau et qui se rencontrent en son milieu.

Exercice 10 : Fréquence

Soit T un tableau contenant n éléments de type entier et x un entier quelconque.
Écrire une fonction **int frequence (int T [] , int n , int x)** qui retourne le nombre d'apparitions de x dans le tableau T.

Exercice 11 : Éléments distincts

Écrire une fonction C qui permet de remplir un tableau T par n entiers différents (chaque entier est présent une seule fois).

Exercice 12 : Nombre d'éléments distincts

On se propose d'écrire un programme C qui saisit un entier n (10<n<=100) puis un tableau T composé de n entiers. Le programme calcule le nombre d'éléments distincts de T.

Exercice 13 : Séquences strictement croissantes

Écrire une fonction C qui permet d'afficher les sous séquences strictement croissantes depuis un tableau de N entiers.
Exemple pour T = 5|7|9|2|3|1|20|25
La fonction affiche :
5|7|9
2|3
1|20|25

Exercice 14: Fréquence

Présenter d'une façon informelle et réaliser en C un algorithme permettant de compter la fréquence des éléments stockés dans un tableau. Ces éléments sont des entiers compris entre 0 et 99.

Exercice 15: Le plus fréquent

Proposer d'une façon informelle et réaliser en C un algorithme qui détermine l'élément le plus fréquent dans un tableau.

TP N°5

Les algorithmes de tri et de recherche

Exercice 1:

L'algorithme de recherche séquentielle (ou linéaire) consiste à examiner la table éléments par éléments et voir si **info** appartient ou non à la table **T**. Si le résultat est positif (**info ∈ T**) alors cet algorithme retourne l'indice de la première occurrence de l'**info**, sinon il retourne -1.

Réaliser en C la fonction **int RechercheSequentielle (int T [] , int n, int info)**

Exercice 2:

Soit le programme suivant :

```
# include <stdio.h>
# define n 5
float a [n] ;
float info ;
unsigned P (unsigned x) {
        return a[x] = = info ;
}
void main(void){
        unsigned x ;
        unsigned y ;
        unsigned p, q ;
        p=0 ;
        q=n ;
        x=p ;
        y=q ;
        while(x !=y) {
                if(P(x))
                        y=x ;
                else
                        x++ ;
        }
        printf("%u\n", x !=q) ;
}
```

Questions :

a) Expliquer l'organisation générale du programme cité ci-dessus.

b) Identifier le problème censé être résolu par le programme précédent et comparer la solution proposée par rapport aux algorithmes résolvant le même type du problème.

Exercice 3:

La technique de tri par sélection est la technique la plus simple, elle consiste à :

- chercher l'indice du plus petit élément du tableau T[0..n-1] et permuter l'élément correspondant avec l'élément d'indice 0
- chercher l'indice du plus petit élément du tableau T[1..n-1] et permuter l'élément correspondant avec l'élément d'indice 1
- …
- chercher l'indice du plus petit élément du tableau T[n-2..n-1] et permuter l'élément correspondant avec l'élément d'indice (n-2).

1) Écrire la fonction **void TriSelection (int T [] , int n)** qui permet le tri des n premiers éléments du tableau d'entier T.
2) Écrire un programme principal (la fonction main) qui lit la dimension n d'un tableau T de type int (dimension maximale: 50 composantes), remplit ce tableau par des valeurs entrées au clavier et affiche le tableau. Ensuite, trier ce tableau en utilisant la fonction TriSelection puis réafficher ce tableau.

Exercice 4:

Cette méthode consiste à prendre les éléments du tableau un par un et insérer chacun dans sa bonne place de façon que les éléments traités forment une sous-liste triée.
Pour ce faire, on procède de la façon suivante :
- comparer et permuter si nécessaire T[0] et T[1] de façon à placer le plus petit dans la case d'indice 0
- comparer et permuter si nécessaire l'élément T[2] avec ceux qui le précèdent dans l'ordre (T[1] puis T[0]) afin de former une sous-liste triée T[0..2]
- …
- comparer et permuter si nécessaire l'élément T[n-1] avec ceux qui le précèdent dans l'ordre (T[n-2], T[n-3], …) afin d'obtenir un tableau trié.

1) Écrire la fonction **void TriInsertion (int T [] , int n)** qui permet le tri des n premiers éléments du tableau d'entier T.
2) Écrire un programme principal (la fonction main) qui lit la dimension n d'un tableau T de type int (dimension maximale: 50 composantes), remplit ce tableau par des valeurs entrées au clavier et affiche le tableau. Ensuite, trier ce tableau en utilisant la fonction TriInsertion puis réafficher ce tableau.

Exercice 5:

Comparer les deux algorithmes tri par sélection et tri par insertion.

Exercice 6 :

Réaliser en C un algorithme qui permet de calculer le nombre de sous-séquences et la plus longue sous-séquences strictement croissante dans un tableau de n entiers.

Exercice 7 :

On donne un tableau appelé redondant contenant des entiers redondants et dans un ordre quelconque.

Présenter d'une façon informelle et réaliser en C un algorithme permettant de compter la fréquence de chaque élément figurant dans le tableau redondant dans une table.

Exercice 8 :

On possède un tableau de nombres entiers, classés par valeur croissante, chaque entier pouvant être répété plusieurs fois : 3 5 5 5 7 9 9 9 9 10 10 12 12 12 13

On appelle plateau la suite de tous les entiers consécutifs de même valeur : dans l'exemple ci-dessous le plateau de valeur 5 comporte 3 nombre, celui de valeur 10 comporte 2 etc. On appelle la longueur d'un plateau la valeur commune à tous ses termes. Un tableau a de n termes étant donné, trouver la longueur et la valeur de son plus long plateau.

Exercice 9:

On donne une table contenant des couples (x, y) avec x est une information de type réel et y sa fréquence (information de type entier non signé). Une telle table est triée par ordre croissant sur le champ x. De plus, on définit les opérations suivantes :

1. cherche : elle permet de voir si une information (info) de type réel appartient ou non à la table. En cas d'issue positive, cherche rend également la position de cet élément. Programmer en C cette opération.
2. incrémenter : elle permet d'incrémenter la fréquence relative à x déjà localisée. Programmer en C cette opération.
3. ajouter : elle permet d'ajouter x avec une fréquence nulle à la table. Elle laisse la table triée par ordre croissant sur le champ x.
   Donner la signature de cette opération et d'une façon informelle l'algorithme adéquat à la programmation de cette opération.

Exercice 10:

Soit T un tableau de n entiers. La méthode de tri à bulles nécessite deux étapes :

- Parcourir les éléments du tableau de 0 à (n–2) ; si l'élément i est supérieur à l'élément (i+1), alors on les permute.
- Le programme s'arrête lorsqu'aucune permutation n'est réalisable après un parcours complet du tableau.

1) Écrire la fonction **void TriBulles (int T [] , int n)** qui permet le tri des n premiers éléments du tableau d'entier T.

2) Écrire un programme principal (la fonction main) qui lit la dimension N d'un tableau T de type int (dimension maximale: 50 composantes), remplit ce tableau par des valeurs entrées au clavier et affiche le tableau. Ensuite, trier ce tableau en utilisant la fonction TriBulles puis réafficher ce tableau.

Exercice 11:

La technique de recherche dichotomique n'est applicable que si le tableau est déjà trié (par exemple dans l'ordre croissant). Le but de recherche dichotomique est de diviser l'intervalle de recherche par 2 à chaque itération. Pour cela, on procède de la façon suivante :

Soient premier et dernier les extrémités gauche et droite de l'intervalle dans lequel on cherche la valeur x, on calcule M, l'indice de l'élément médian :

M = (premier + dernier) div 2

Il y a 3 cas possibles :

- x = T[M] : l'élément de valeur x est trouvé, la recherche est terminée
- x < T[M] : l'élément x, s'il existe, se trouve dans l'intervalle [premier..M-1]
- x > T[M] : l'élément x, s'il existe, se trouve dans l'intervalle [M+1..dernier]

La recherche dichotomique consiste à itérer ce processus jusqu'à ce que l'on trouve x ou que l'intervalle de recherche soit vide.